(1)

AFIT/GE/ENG/89M-6

FLIGHT CONTROL SYSTEM FOR THE CRCA USING A COMMAND GENERATOR TRACKER WITH PI FEEDBACK AND KALMAN FILTER VOLUME II

THESIS

Steven Spencer Payson
Captain, USAF

AFIT/GE/ENG/89M-6

AFIT/GE/ENG/89M-6

FLIGHT CONTROL SYSTEM FOR THE CRCA USING A COMMAND GENERATOR TRACKER WITH PI FEEDBACK AND KALMAN FILTER VOLUME II

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

Steven Spencer Payson, B.S.
Captain, USAF

March, 1989

Acknowledgments

I'd like to express my deepest thanks to Dr. Peter S. Maybeck for his concern, dedication, and patience. Without his guidance, this thesis effort would not have been the outstanding learning experience it turned out to be.

I would also like to thank Daryl Hammond and Kurt Neumann for helping me understand the CRCA. Dan Zambon provided invaluable assistance as the resident computer guru. Tom Kobylarz provided some valuable insights into MATRIX$_X$. His parties, along with the efforts of Chuck Sokol and Dave Rizzo, helped me keep my perspective. To my climbing buddies, thanks for keeping me on the rock, and not letting go of the rope.

Finally, I'd like to thank Bob Dudley, my Little Father, for a ray of the sun source, and Tim Sakulich, for my sanity.

Steven Spencer Payson

Table of Contents

Page

AFIT/GE/ENG/89M-6

Abstract

This research develops an integrated software design package useful in the synthesis of CGT/PI/KF control systems, and uses this software package to design and evaluate a longitudinal flight control system for the Control Reconfigurable Combat Aircraft (CRCA). The software package, called CGTPIKF and built with MATRIX$_X$ commands, allows for the synthesis and evaluation of a Command Generator Tracker (CGT) which provides inputs to the system and acts as a precompensator, and a regulator with proportional plus integral (PI) feedback which forces the system outputs to mimic the model output. The software also allows the incorporation of a Kalman filter for estimation of the system states. Certainty equivalence can be invoked by adopting the LQG assumptions, thereby allowing the Kalman filter to be designed independently of the CGT/PI controller. The total CGT/PI/KF controller can then be evaluated and the design refined. CGTPIKF is an interactive, menu-driven CAD package which can be used in the development of any CGT/PI/KF control system, regardless of application.

A flight control system was designed for the CRCA air combat mode (ACM) entry using CGTPIKF. This control system was designed to force the aircraft to emulate a first order responce in pitch rate. The command model of the command generator tracker represented a first order pitch rate response with a rise time of .6 sec. Various weighting matrices were evaluated and refined in the development of the PI controller; the different controller designs were tested against the simulation containing various modelling errors, particularly failure conditions. The Kalman filter was later added, and the controller was again tested against the failure conditions. Loop Transmission Recovery (LTR) was successfully implemented to enhance robustness. The results confirm that a robust control system can be designed using the software package developed in this research.

Appendix B. *CGTPIKF Code*

This Appendix presents all of the CGTPIKF code. This code is self documenting, so this Appendix may be considered a programming guide. Refer to Appendix A for the program hierarchy.

B.1 Top Level CGTPIKF Command File

```
// CGTPIKF command file
//
//*******************************************************************
//
// This command file is for the overall control of Payson's thesis
//  software.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (6 Jan)
//*******************************************************************
//
ERASE // Clear the screen
//
// Define a menu for the appropriate tasks.
//
TOPMENU = ['CGTPIKF TOP MENU '
           '  INPUT MODELS   '
           'CGT/PI CONTROLLER'
           '  KALMAN FILTER  '
           'ANALYZE CGT/PI/KF'
           '      QUIT       '];
```

```
//
// Now use the menu
//
K=0; // K is the menu input parameter
WHILE K<>5,...
 K=MENU(TOPMENU);...
 IF K=1, EXECUTE ('MODELINPUTS.'),END,...
 IF K=2, EXECUTE ('CGTPI.'),END,...
 IF K=3, EXECUTE('KF.'),END,...
 IF K=4, EXECUTE('CGTPIKF.ANALYZE'),END,...
END
//
DISP('IF YOU WISH TO SAVE THE CONTENTS OF THE STACK INTO A FILE, ENTER')
DISP('A  0. OTHERWISE, ENTER 1. REGARDLESS OF WHAT YOU ENTER, ALL')
DISP('VARIABLES WILL REMAIN ON THE STACK FOR YOU TO DO WITH AS YOU')
DISP('PLEASE IN THE MAIN MATRIXX PROGRAM ENVIRONMENT.')
INQUIRE TOPMENU 'ENTER A  0 OR 1   '
IF TOPMENU=0,...
  EXECUTE('SAVEFILE.'),...
END
CLEAR TOPMENU K
//
// End OVERALL command file
//
RETURN
```

B.2 MODELINPUTS and Associated Sub-files

```
// MODELINPUTS Command File
//
//*********************************************************************
//
// This command file allows for inputting the design model, command
//  model, and noise model. In reality, the noise model is part of
//  the design model.
//
// Variable Definitions:  ADM = Design model A matrix
//                        BDM = Design model B matrix
//                        CDM = Design model C matrix
//                        DDM = Design model D matrix
//                         EX = Design model Ex matrix
//                         EY = Design model EY matrix
//                        ACM = Command model A matrix
//                        BCM = Command model B matrix
//                        CCM = Command model C matrix
//                        DCM = Command model D matrix
//                        ANM = Noise model A matrix
//                        GNM = Noise model G matrix
//                        SDM = [ADM,BDM;CDM,DDM]
//                        SCM = [ACM,BCM;CCM,DCM]
//                       NSDM = Number of states of design model
//                       NSCM = Number of states of command model
//                       NSNM = Number of states of noise model
//                    NINPUTS = Number of design and command model
//                              inputs are equal and equal # of outputs
```

```
//
// The MODELINPUTS command file initializes and calls three user defined
//  functions, DESMODEL, COMMODEL, and NOISEMOD, which interactively
//  prompt the user for the necessary model matrices.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (19 Jan)
//*************************************************************************
//
DISP('THE DESIGN, COMMAND, AND NOISE MODELS, AND SAMPLE TIME DT,')
DISP('MUST BE INPUT, OR THE PROGRAM WILL ABORT.  THE DESIGN MODEL MUST
BE')
DISP('INPUT BEFORE THE COMMAND, NOISE, TRUTH, OR IMPLICIT MODELS.')
DISP('THE TRUTH MODEL DOES NOT HAVE TO BE INPUT UNTIL YOU')
DISP('EVALUATE YOUR CONTROLLER, AND THE IMPLICIT MODEL MAY BE INPUT')
DISP('AT A LATER TIME. THIS PROGRAM DOES NOT PRESENTLY USE THE NOISE')
DISP('MODEL, BUT IT STILL MUST BE INPUT.')
PAUSE
DISP(' ')
DISP('FOR ALL MODELS, IF A MATRIX ALREADY EXISTS AND YOU DO NOT WANT')
DISP('TO CHANGE IT, JUST TYPE IN THE APPROPRIATE MATRIX NAME (EG, ADM).')
DISP(' '),PAUSE
// Define a menu for the appropriate functions
//
MODMENU = [' MODEL MENU   '
           '  LOAD FILE   '
           ' DESIGN MODEL '
           ' NOISE MODEL  '
```

```
            'COMMAND MODEL '
            ' TRUTH MODEL  '
            'IMPLICIT MODEL'
            ' SAMPLE TIME  '
            ' SAVE MODELS  '
            '    QUIT      '];
//
// Now use the menu
//
OPTION=0;
WHILE OPTION<>9,...
 OPTION=MENU(MODMENU);...
 IF OPTION=1,...
   EXECUTE('INPUTFILE.');END,...
 IF OPTION=2,...
   EXECUTE ('DESMODEL.');END,...
 IF OPTION=3,...
   EXECUTE ('NOISEMOD.');END,...
 IF OPTION=4,...
   EXECUTE ('COMMODEL.');END,...
 IF OPTION=5,...
   EXECUTE ('TRUMODEL.');END,...
 IF OPTION=6,...
   EXECUTE('IMPLICIT.MODEL');END,...
 IF OPTION=7,...
   INQUIRE DT 'SAMPLE TIME (DT)=',END,...
 IF OPTION=8,...
   DISP('IF YOU WANT TO SAVE EVERYTHING ON THE STACK, TYPE A 0.'),...
   DISP('OTHERWISE, TYPE A 1.'),...
   INQUIRE SAVEOPTION 'OPTION IS (0 OR 1) ',...
   IF SAVEOPTION=0, OPTION=0;...//Otherwise if you load a file, you'll
        ...// come back to this option because OPTION=7 in the saved file
   EXECUTE('SAVEFILE.'),END,...
 END,...
END
//
CLEAR OPTION MODMENU // To reduce the number of variables being passed
around
CLEAR SAVEOPTION
//
// End of MODELINPUTS command file
//
RETURN
```

```
//INPUTFILE Command file
//
//***************************************************************************
//
// This command file allows the user to input a file name as a text string.
//  This command file is called by the MODELINPUTS and CGTPI.ANALYZE
//  command files.  The reason it is a seperate command file is because
//  MATRIXX threw up on me when I tried including these commands in Option
//  1 of the MODELINPUTS command file.  As a seperate command file
//  (INPUTFILE.SUB), however,these commands work.
//
// This command file calls the INPUTFILE.SUB command file.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M.
// Version 1.0  (6 Jan)
//***************************************************************************
//
// Tell the user how to use this program
//
DISP('IF YOU WISH TO LOAD A FILE FROM MEMORY, TYPE THE FILENAME WITH')
DISP('SINGLE QUOTATION MARKS AROUND THE FILENAME.')
//
// Tell the user what the potential problems of reading in new files
//  are, and give them an opportunity to back out.
//
DISP('***************** WARNING   WARNING   WARNING ****************')
DISP('IF THE FILENAME DOES NOT EXIST, YOU WILL BE BOOTED RIGHT OUT OF')
DISP('THIS ENTIRE PROGRAM, BACK TO MATRIXX.  ALSO, THE FILE YOU LOAD MAY')
DISP('OVERWRITE EXISTING VARIABLES ON THE STACK, SO BE CAREFUL.')
DISP('ENTER A  0 <ZERO> IF YOU WISH TO CONTINUE.  OTHERWISE, TYPE A 1.')
INQUIRE DUMMY 'OPTION:  0  OR  1     '
//
IF DUMMY=0, EXECUTE('INPUTFILE.SUB'),END
CLEAR DUMMY
//
// End of INPUTFILE command file
//
RETURN




// DESMODEL command file
//
//*******************************************************************
// The DESMODEL command file interactively prompts the user to input
//  the design model. The design model is of the form:
//     dx/dt = A x + B u +Ex n + Gw
//            y = C x + D u + Ey n  (Output equation)
//     n = An n + Gn w (Which is input in another command file)
//            z = H x + v     (Measurement equation)
//
// The noise model is not presently used in the CGTPIKF
//  command file software.
//
// Variable Definitions:  ADM = Design model (DM) A matrix
//                              BDM = Design model B matrix
//                              CDM = Design model C matrix
//                              DDM = Design model D matrix
//                               EX = Design model Ex matrix
//                               EY = Design model Ey matrix
//                              SDM = [ADM,BDM;CDM,DDM]
//                              GDM = Design model dynamics noise input matrix
//                              HDM = Design model measurement vector
//                             NSDM = Number of states of design model
//                             NSNM = Number of states of noise model
//                          NINPUTS = Number of DM inputs and outputs
//                            NMEAS = Number of measurements
//
// The DUMMY variable is used in conditional loops to allow the
//  user to fix blatant mistakes made while inputting each matrix.
//  Blatant mistakes are those involving dimensionality problems,
//  and the user is given five chances to fix these. Changing matrices
//  for other than blatant mistakes can be done in the INPUTS command
//  function.
//
// This command file is called by the MODELINPUTS command file.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (9 Jan)
//*******************************************************************
//
DISP('YOU ARE ABOUT TO ENTER THE CONTINUOUS DESIGN MODEL. IT IS OF THE
FORM')
```

```
DISP('dx/dt = A*x + B*u + Ex*n + G*w')
DISP('y = C*x + D*u + Ey*n')
DISP('dn/dt = An*n + Gn*w (WHICH WILL BE INPUT LATER)')
DISP('Z = H*X + V ')
DISP('WHERE W HAS STRENGTH Q AND V HAS STRENGTH R.')
PAUSE
DISP('THE STRENGTHS OF NOISES W AND V WILL BE INPUT LATER.')
DISP('THE NOISE MODEL N(TI+1) IS NOT PRESENTLY USED IN THIS SOFTWARE.')
DISP('THE EX AND EY MATRICES ARE SET TO ZERO LATER.')
DISP(' ')
DISP('NOW ENTER THE A,B,C,D,EX,EY,G AND H MATRICES OF THE DESIGN MODEL')
//
// Input A matrix
//
DUMMY=0;
WHILE DUMMY<1,...
 INQUIRE ADM 'A MATRIX (ADM)=',...
 [NS,N]=SIZE(ADM); IF NS=N, DUMMY=1;...
   ELSE DISP('ADM MUST BE SQUARE. START AGAIN'),END,...
END
NSDM=N; // NSDM is Number of States of Design Model
//
// NS = N = number of states
//
// Input B matrix. Number of rows M must equal the number of A matrix
//  rows (and thus the number of states).
//
DUMMY=0;
```

```
WHILE DUMMY<1,...
 INQUIRE BDM 'B MATRIX (BDM)=',...
 [M,N]=SIZE(BDM); IF M=NS,DUMMY=1;...
   ELSE DISP('NUMBER OF ROWS OF BDM MUST EQUAL THE NUMBER OF STATES'),NSDM,..
   END,...
 NINPUTS=N;...
END
//
// Input C matrix. Number of columns Q must equal the number of
//  A matrix columns (and thus the number of states).
//
DUMMY=0;
WHILE DUMMY<1,...
 INQUIRE CDM 'C MATRIX (CDM)=',...
 [P,Q]=SIZE(CDM); IF Q<>NS,...
   DISP('NUMBER OF COLUMNS OF CDM MUST EQUAL THE NUMBER OF STATES'),NSDM,...
   ELSE IF P=N, DUMMY=1;...// Rows of CDM equal columns of BDM
     ELSE DISP('NUMBER OF OUTPUTS MUST EQUAL NUMBER OF INPUTS'),N,...
 END,END,...
END
//
// Input D matrix. Number of rows R must match the number of rows of C
matrix,
//  and the number of columns must equal the number of columns of the
B
//  matrix.
//
DUMMY=0;
```

```
WHILE DUMMY<1,...
 INQUIRE DDM 'D MATRIX (DDM) (YOU MAY ENTER A ZERO) = ',...
...//
...// If user inputs a 0, make DDM the correct dimensionality
...//
 [R,S]=SIZE(DDM); IF R=1, IF S=1,... //Can't do next step if DDM isn't
a scalar
    IF DDM=0, DDM=0*ONES(P,N),END,...
 END,END,...
...//
...// Ensure DDM is the correct dimensionality. If a 0 was input, it already
...//  will be, but we'll check anyway.
...//
 [R,S]=SIZE(DDM); IF R<>P,...
   DISP('NUMBER OF ROWS OF DDM MUST EQUAL THE NUMBER OF ROWS OF CDM'),N,...
   ELSE IF S=N,DUMMY=5;...
     ELSE DISP('NUMBER OF COLUMNS OF DDM MUST EQUAL NUMBER OF COLUMNS'),N,...
     DISP('OF BDM,THAT IS, ACCOMODATE ALL INPUTS'),...
 END,END,...
 DUMMY=DUMMY+1;...
END
//
// Input the Ex matrix, which is the time correlated noise matrix. The
number
//  of rows R must equal the number of A matrix rows (and thus the number
//  of states). The number of columns of Ex equals the number of noise
model
//  states, NSNM.
```

```
//
DUMMY=0;
WHILE DUMMY<1,...
 INQUIRE EX 'EX MATRIX (YOU MAY ENTER A ZERO) = ',...
...//
...// If user inputs a 0, make EX the correct dimensionality
...//
 [R,S]=SIZE(EX); IF R=1, IF S=1,...//Can't do next step if EX isn't a
scalar
    IF EX=0,EX=0*ONES(NS,1),END,...
 END,END,...
...//
...// Ensure EX is the correct dimensionality. If a 0 was input, it already
...//  will be, but we'll check anyway.
...//
 [R,NSNM]=SIZE(EX); IF R=NS,DUMMY=1;...
   ELSE DISP('THE NUMBER OF ROWS OF EX MUST EQUAL'),...
   DISP('THE NUMBER OF STATES OF THE SYSTEM'),NSDM,END,...
END
//
// Input the Ey matrix. The number of rows R must equal the number of
//  rows P of the C matrix, and the number of columns S must equal the
//  number of noise model states, NSNM.
//
DUMMY=0;
WHILE DUMMY<1,...
 INQUIRE EY 'EY MATRIX (YOU MAY ENTER A ZERO) = ',...
...//
```

```
...// If user inputs a 0, make EY the correct dimensionality
...//
 [R,S]=SIZE(EY); IF R=1, IF S=1,...//Can't do next step if EY isn't a
scalar
    IF EY=0,EY=0*ONES(P,NSNM),END,...
 END,END,...
...//
...// Ensure EY is the correct dimensionality. If a 0 was input, it already
...//  will be, but we'll check anyway.
...//
 [R,S]=SIZE(EY); IF R<>P,...
   DISP('THE NUMBER OF ROWS OF EY MUST EQUAL'),...
   DISP('THE NUMBER OF ROWS OF THE C MATRIX'),N,...
   ELSE IF S=NSNM, DUMMY=1;ELSE...
     DISP('NUMBER OF COLUMNS OF EY MUST EQUAL THOSE OF EX'),...
 END,END,...
END
//
// Enter the G matrix.  It must be square with dimensions equal to the
//  number of states, NSDM.
//
DUMMY=0;
WHILE DUMMY<1,...
 DISP('NOW ENTER THE G MATRIX FOR ENTRY OF DYNAMICS NOISES.  IF YOU ARE'),...
 DISP('PURSUING A DETERMINISTIC CONTROL SYSTEM, YOU MAY TYPE A ZERO.'),...
 DISP('THE G MATRIX MUST HAVE THE SAME NUMBER OF ROWS AS'),...
 NSDM, INQUIRE GDM 'UNLESS YOU INPUT A ZERO.  GDM =  ',...
...//
```

```
...// If user inputs a 0, make GDM the correct dimensionality
...//
 [R,S]=SIZE(GDM); IF R=1, IF S=1,...//Can't do next step if GDM isn't
a scalar
    IF GDM=0,GDM=0*ONES(NSDM,1),END,...//Assume GDM a column vector
 END,END,...
...//
...// Ensure GDM is the correct dimensionality. If a 0 was input, it already
...//  will be, but we'll check anyway.
...//
 [R,S]=SIZE(GDM); IF R=NSDM, DUMMY=1;...
   ELSE DISP('WRONG NUMBER OF ROWS. TRY AGAIN.'),...
 END,...
END
//
DISP('NOW ENTER THE H MATRIX.  THE H MATRIX IS THE SYSTEM MEASUREMENT')
DISP('MATRIX.  IT MAY BE THE SAME AS THE C MATRIX.  IF THE MEASUREMENT')
DISP('MATRIX IS THE SAME AS THE OUTPUT MATRIX, JUST TYPE CDM.')
DUMMY=0;
WHILE DUMMY<1,...
 INQUIRE HDM 'H MATRIX (HDM) = ',...
 [P,Q]=SIZE(HDM); IF Q<>NS,...
   DISP('NUMBER OF COLUMNS OF HDM MUST EQUAL THE NUMBER OF STATES'),NSDM,...
   ELSE DUMMY=1; END,...
 NMEAS = P;...//NMEAS is the number of measurements
END
//
// Form the system matrix SDM.
```

```
//
SDM=[ADM,BDM;CDM,DDM];NSDM=NS;
//
// Now display the design model to the user.
//
DISP('THE DESIGN MODEL IS (HIT RETURN KEY AFTER EACH DISPLAY):')
ADM,PAUSE,BDM,PAUSE,EX,PAUSE,CDM,PAUSE,DDM,PAUSE,EY,PAUSE,GDM
PAUSE,HDM,PAUSE
//
// End of DESMODEL command file
//
CLEAR DUMMY NS M N P Q R S //Clear extraneous variables
RETURN
```

```
// NOISEMOD command file
//
//*********************************************************************
// The NOISEMOD command file interactively prompts the user to input
//  the noise model. The noise model is of the form:
//      n(Ti+1) = An n(Ti) + Gn w(Ti)
//
// In reality, the noise model is part of the design model.
//
// Variable Definitions:  ANM = Noise model A matrix
//                        GNM = Noise model G matrix
//                       NSNM = Number of states of the noise model,
//                              equal to number of columns of Ex in
//                              design model
//
//
// The DUMMY variable is used in conditional loops to allow the
//  user to fix blatant mistakes made while inputting each matrix.
//  Blatant mistakes are those involving dimensionality problems,
//  and the user is given five chances to fix these. Changing matrices
//  for other than blatant mistakes can be done in the INPUTS command
//  file.
//
// This command file is called by the INPUTS command file.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (14 Oct)
// The noise model is presently not used anywhere in the OVERALL
//  software.  This will be an option for later versions of this software
//*********************************************************************
//
DISP('YOU ARE ABOUT TO ENTER THE NOISE MODEL, WHICH IS PART OF')
DISP('THE DESIGN MODEL. IT IS OF THE FORM')
DISP('n(Ti+1) = An*n(Ti) + Gn*w(Ti)')
DISP(' ')
DISP('NOW ENTER THE A AND G MATRICES OF THE NOISE MODEL')
//
// Input A matrix (ANM). ANM must be square and have dimensions of NSNM,
//  which is equal to the number of columns of Ex in the design model.
//
DUMMY=1;
WHILE DUMMY<5,...
 INQUIRE ANM 'A MATRIX (ANM)=',...
 [R,S]=SIZE(ANM); IF R<>S, DISP('ANM MUST BE SQUARE. START AGAIN'),...
   ELSE IF R=NSNM, DUMMY=5;...
     ELSE DISP('NUMBER OF STATES OF NOISE MODEL MUST EQUAL NUMBER'),...
     DISP('OF ROWS OF EX'),NSNM,...
 END,END,...
 DUMMY=DUMMY+1;...
END
//
// NSNM = number of states of noise model
//
// Input G matrix (GNM). Number of rows R must equal the number of A matrix
//  rows (and thus the number of noise model states).
//
```

```
DUMMY=1;
WHILE DUMMY<5,...
 INQUIRE GNM 'G MATRIX (GNM)=',...
 [R,S]=SIZE(GNM); IF R=NSNM,DUMMY=5;...
   ELSE DISP('NUMBER OF ROWS OF GNM MUST EQUAL THE NUMBER OF STATES'),END,...
 DUMMY=DUMMY+1;...
END
//
// Now display the noise model to the user.
//
DISP('THE NOISE MODEL IS (HIT RETURN KEY AFTER EACH DISPLAY):')
ANM,PAUSE,GNM,PAUSE
CLEAR DUMMY R S // Clear extraneous variables
//
// End of NOISEMOD command file
//
RETURN
```

```
// COMMODEL command file
//
//*********************************************************************
// The COMMODEL command file interactively prompts the user to input
//  the command model. The command model is of the form:
//        dx/dt = A x + B u
//            y = C x + D u
//
// Variable Definitions:  ACM = Command model A matrix
//                        BCM = Command model B matrix
//                        CCM = Command model C matrix
//                        DCM = Command model D matrix
//                        SCM = [ACM,BCM;CCM,DCM]
//                       NSCM = Number of states of command model
//                        CDM = Design model C matrix
//
// The DUMMY variable is used in conditional loops to allow the
//  user to fix blatant mistakes made while inputting each matrix.
//  Blatant mistakes are those involving dimensionality problems,
//  and the user is given five chances to fix these. Changing matrices
//  for other than blatant mistakes can be done in the MODELINPUTS command
//  file.
//
// This command file is called by the MODELINPUTS command file.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (19 Jan)
//*********************************************************************
```

```
//
DISP('THE DESIGN MODEL MUST BE INPUT BEFORE INPUTTING THE COMMAND MODEL')
DISP('YOU ARE ABOUT TO ENTER THE COMMAND MODEL. IT IS OF THE FORM')
DISP('dx/dt = A*x + B*u')
DISP('y = C*x + D*u')
DISP('WHERE A, B, C, AND D ARE CONTINUOUS TIME MATRICES. ')
DISP(' ')
DISP('NOW ENTER THE A,B,C,AND D MATRICES OF THE COMMAND MODEL')
//
// Input A matrix (ACM)
//
DUMMY=1;
WHILE DUMMY<5,...
 INQUIRE ACM 'A MATRIX (ACM)=',...
 [NS,N]=SIZE(ACM); IF NS=N, DUMMY=5;...
   ELSE DISP('ACM MUST BE SQUARE. START AGAIN'),END,...
 DUMMY=DUMMY+1;...
END
NSCM=N;
//
// NS = N = number of states
//
// Input B matrix (BCM). Number of rows M must equal the number of A matrix
//  rows (and thus the number of states), and columns must equal the number
//  of design model inputs, NINPUTS.
//
DUMMY=0;
WHILE DUMMY<1,...
```

```
 DISP('THE COMMAND MODEL MUST HAVE THE SAME NUMBER OF INPUTS AND OUTPUTS'),..
 DISP('AS THE DESIGN MODEL.'),NINPUTS,...
 INQUIRE BCM 'B MATRIX (BCM)=',...
 [M,N]=SIZE(BCM); IF M<>NS,...
   DISP('NUMBER OF ROWS OF BCM MUST EQUAL THE NUMBER OF STATES'),NSCM,...
  ELSE IF N=NINPUTS, DUMMY=1;...
     ELSE DISP('WRONG NUMBER OF COLUMNS.  TRY AGAIN.'),...
 END,END,...
END
//
// Input C matrix (CCM). Number of columns Q must equal the number of
//  A matrix columns (and thus the number of states).
//
DUMMY=0;
WHILE DUMMY<1,...
 DISP('NUMBER OF ROWS OF CCM MUST EQUAL THE NUMBER OF ROWS OF CDM'),NINPUTS,.
 INQUIRE CCM 'C MATRIX (CCM)=',...
 [P,Q]=SIZE(CCM); IF Q<>NS,...
   DISP('NUMBER OF COLUMNS OF CCM MUST EQUAL THE NUMBER OF STATES'),NSCM,...
  ELSE IF P=NINPUTS, DUMMY=1;...
     ELSE DISP('WRONG NUMBER OF ROWS.  TRY AGAIN.'),...
 END,END,...
END
//
// Input D matrix (DCM). Number of rows R must match the number of rows
//  of C matrix, and the number of columns must equal the number oF
//  columns of the B matrix.
//
```

```
DUMMY=1;
WHILE DUMMY<5,...
 INQUIRE DCM 'D MATRIX (DCM)=',...
...//
...// If user inputs a 0, make DCM the correct dimensionality. If a 0
was
...//  input, it already will be, but we'll check anyway.
...//
 [R,S]=SIZE(DCM); IF R=1, IF S=1,... //Can't do next step if DCM isn't
a scalar
    IF DCM=0, DCM=0*ONES(P,N),END,... //This makes DCM the right dimensionali
 END,END,...
...//
...// Ensure DCM is the correct dimensionality
...//
 [R,S]=SIZE(DCM); IF R=P,DUMMY=5;...
   ELSE DISP('NUMBER OF ROWS OF DCM MUST EQUAL THE NUMBER OF ROWS OF CCM'),..
   NINPUTS, END,...
 IF S=N,DUMMY=5;...
   ELSE DISP('NUMBER OF COLUMNS OF DCM MUST EQUAL NUMBER OF COLUMNS'),...
    DISP('OF BCM,THAT IS, ACCOMODATE ALL INPUTS'),NINPUTS,...
 END,...
 DUMMY=DUMMY+1;...
END
//
// Form the system matrix
//
SCM=[ACM,BCM;CCM,DCM];NSCM=NS;
```

```
//
// Now display the command model to the user.
//
DISP('THE COMMAND MODEL IS (HIT RETURN KEY AFTER EACH DISPLAY):')
ACM,BCM,PAUSE,CCM,DCM,PAUSE
CLEAR DUMMY R S NS N M P Q // Clear extraneous variables
//
// End of COMMODEL command file
//
RETURN
```

```
// TRUMODEL command file
//
//*********************************************************************
// The TRUMODEL command file interactively prompts the user to input
//  the truth model. The truth model is of the form:
//      dx/dt  = A x + B u + G*w
//           y = C x + D u
//           z = H x + v
//
// Variable Definitions:  ATM = Truth model (TM) A matrix
//                        BTM = Truth model B matrix
//                        CTM = Truth model C matrix
//                        DTM = Truth model D matrix
//                        STM = [ATM,BTM;CTM,DTM]
//                       NSTM = Number of states of design model
//                        GDM = Truth model G (noise input) matrix
//                        HDM = Truth model H (measurement) matrix
//                  TMNINPUTS = Number of TM inputs and outputs
//                    TMNMEAS = Number of TM masurements
//
// The DUMMY variable is used in conditional loops to allow the
//  user to fix blatant mistakes made while inputting each matrix.
//  Blatant mistakes are those involving dimensionality problems,
//  and the user is given five chances to fix these. Changing matrices
//  for other than blatant mistakes can be done in the INPUTS command
//  function.
//
```

```
// This command file is called by the MODELINPUTS command file.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (31 Jan)
//*******************************************************************
//
DISP('YOU ARE ABOUT TO ENTER THE CONTINUOUS TRUTH MODEL. IT IS OF THE
FORM')
DISP('DX/DT = A*x + B*u + G*w')
DISP('y = C*x + D*u ')
DISP('Z = H*X + V')
DISP('THE TRUTH MODEL DOES NOT HAVE EX OR EY MATRICES AS THE DESIGN ')
DISP('MODEL DOES BECAUSE THEY SHOULD BE INCORPORATED DIRECTLY INTO THE')
DISP('TRUTH MODEL. NOTE THAT THESE ARE CONTINUOUS-TIME MATRICES.')
PAUSE
DISP('NOW ENTER THE A,B,C,D,G,H,Q,R,AND PTO MATRICES OF THE TRUTH MODEL.')
DISP('(THE TM ON THE MATRIX NAMES REFERS TO THE FACT THAT YOU ARE')
DISP('INPUTTING THE TRUTH MODEL MATRICES.)')
DISP('WHEN INCLUDING ACTUATOR DYNAMICS IN THE TRUTH MODEL, MAKE THE')
DISP('ACTUATOR STATES THE LAST STATES ENTERED, STARTING WITH THE LOWEST')
DISP('DERIVATIVE; THE HIGHEST DERIVATIVE OF THE ACTUATORS WILL BE THE
LAST')
DISP('TRUTH MODEL STATES.  THE NUMBER OF STATES PER ACTUATOR MUST BE THE')
DISP('SAME, SO THE DERIVATIVES MUST ALTERNATE BETWEEN THE ACTUATORS.')
DISP('FOR MORE DETAILS, SEE THE USERS MANUAL.')
//
// Input A matrix
//
```

```
DUMMY=0;
WHILE DUMMY<1,...
 INQUIRE ATM 'A MATRIX (ATM)=',...
 [NS,N]=SIZE(ATM); IF NS=N, DUMMY=1;...
   ELSE DISP('ATM MUST BE SQUARE. START AGAIN'),END,...
END
NSTM=NS;//NSTM is number of states of the truth model
INQUIRE NSACT 'ENTER THE NUMBER OF STATES OF THE ACTUATORS, NSACT='
//
// NS = N = number of states
//
// If NSTM is different from NSDM, prompt the user for a transformation.
//  This is required for the simulation, as the PI gain KX is tied to
the
//  design model dimensions (see GAIN1.BLOCKS command file).
//
IF NSTM<>NSDM,...
  DUMMY=0;...
  WHILE DUMMY<1,...
   DISP('BECAUSE THE NUMBER OF STATES OF THE TRUTH MODEL'),...
   DISP('IS DIFFERENT FROM THE NUMBER OF STATES OF THE DESIGN MODEL, YOU'),..
   DISP('MUST ENTER A TRANSFORMATION MATRIX (TTM) WHICH DESCRIBES HOW'),..
   DISP('THE TRUTH MODEL STATES MAP INTO THE DESIGN MODEL STATES.'),...
   DISP('THIS TRANSFORMATION MATRIX MUST HAVE THE FOLLOWING NUMBER OF'),...
   DISP('AND COLUMNS:'),ROWS=NSDM, COL=NSTM,...
   INQUIRE TTM 'ENTER TTM: ',...
   [M,N]=SIZE(TTM); IF M<>NSDM,...
     DISP('NUMBER OF ROWS OF BDM MUST EQUAL THE NUMBER OF STATES'),...
```

```
    ELSE IF N=NSTM, DUMMY=1;...
      ELSE DISP('WRONG NUMBER OF COLUMNS.  TRY AGAIN. '),...
   END,END,...
  END,... // end of WHILE loop
ELSE TTM=EYE(NSTM),... // TTM matrix must exist; this leaves the states
END                    //  the same, without any transformation
//
// Input B matrix. Number of rows M must equal the number of A matrix
//  rows (and thus the number of states).
//
DUMMY=0;
WHILE DUMMY<1,...
 DISP('THE B MATRIX MUST HAVE THE SAME NUMBER OF COLUMNS AS THE NUMBER'),...
 DISP('OF DESIGN MODEL INPUTS'),NINPUTS,...
 INQUIRE BTM 'B MATRIX (BTM)=',...
 [M,N]=SIZE(BTM); IF M<>NS,...
   DISP('NUMBER OF ROWS OF BDM MUST EQUAL THE NUMBER OF STATES'),...
  ELSE IF N=NINPUTS, DUMMY=1;...
     ELSE DISP('WRONG NUMBER OF COLUMNS.  TRY AGAIN. '),...
 END,END,...
 TMNINPUTS=N;...
END
//
// Input C matrix. Number of columns q must equal the number of
//  A matrix columns (and thus the number of states).
//
DUMMY=0;
WHILE DUMMY<1,...
```

```
 DISP('THE NUMBER OF ROWS OF CTM MUST EQUAL THE NUMBER OF DESIGN'),...
 DISP('MODEL INPUTS (AND OUTPUTS) FOR ADEQUATE EVALUATION LATER. '),NINPUTS,.
 INQUIRE CTM 'C MATRIX (CTM)=',...
 [P,Q]=SIZE(CTM); IF Q<>NS,...
   DISP('NUMBER OF COLUMNS OF CTM MUST EQUAL THE NUMBER OF STATES'),...
  ELSE IF P=NINPUTS, DUMMY=1;...
    ELSE DISP('WRONG NUMBER OF ROWS. TRY AGAIN.'),...
 END,END,TMNOUTPUTS=Q;...
END
//
// Input D matrix. Number of rows R must match the number of rows of C
matrix,
//  and the number of columns must equal the number of columns of the
B
//  matrix.
//
DUMMY=1;
WHILE DUMMY<5,...
 INQUIRE DTM 'D MATRIX (DTM)=',...
...//
...// If user inputs a 0, make DTM the correct dimensionality
...//
 [R,S]=SIZE(DTM); IF R=1, IF S=1,... //Can't do next step if DTM isn't
a scalar
    IF DTM=0, DTM=0*ONES(P,N),END,...
 END,END,...
...//
...// Ensure DTM is the correct dimensionality. If a 0 was input, it already
```

```
...//  will be, but we'll check anyway.
...//
 [R,S]=SIZE(DTM); IF R<>P,...
   DISP('NUMBER OF ROWS OF DTM MUST EQUAL THE NUMBER OF ROWS OF CTM'),...
   ELSE IF S=N,DUMMY=5;...
     ELSE DISP('NUMBER OF COLUMNS OF DTM MUST EQUAL NUMBER OF COLUMNS'),...
     DISP('OF BTM,THAT IS, ACCOMODATE ALL INPUTS'),...
 END,END,...
 DUMMY=DUMMY+1;...
END
//
// Input the G matrix.  It should have the same number of rows as the
//  number of states, NSTM
//
DUMMY=0;
WHILE DUMMY<1,...
 DISP('NOW ENTER THE G MATRIX FOR ENTRY OF DYNAMICS NOISES.  IF THE '),...
 DISP('TRUTH MODEL IS DETERMINISTIC, YOU MAY TYPE A ZERO.  THE G '),...
 DISP('MATRIX MUST HAVE THE SAME NUMBER OF ROWS AS,'),NSTM,...
 INQUIRE GTM 'UNLESS YOU INPUT A ZERO.  GTM =  ',...
...//
...// If user inputs a 0, make GTM the correct dimensionality
...//
 FLAG=0;... // Initialize a flag variable
 [R,S]=SIZE(GTM); IF R=1, IF S=1,...//Can't do next step if GTM isn't
a scalar
    IF GTM=0,GTM=0*ONES(NSTM,1);FLAG=1;END,...//Make GTM a column vector
 END,END,...// FLAG is a flag variable which identifies the truth model
```

```
...//          as being deterministic.
...//
...// Ensure GTM is the correct dimensionality. If a 0 was input, it already
...//  will be, but we'll check anyway.
...//
 [R,S]=SIZE(GTM); IF R<>NSTM, DISP('WRONG NUMBER OF ROWS. TRY AGAIN'),...
   ELSE  DUMMY=1;...
 END,...
END
DISP(' ')
//
// Enter the Q matrix, the system dynamics noise strength.
//
DUMMY=0;
IF FLAG=1, DISP('YOU DO NOT HAVE ANY DYNAMICS NOISE, SO Q WONT BE USED'),...
 DISP(' '),QTM=0*EYE(S);DUMMY=1;...// QTM is square with dim of GDM columns
END
 WHILE DUMMY<1,...
  DISP('NOW ENTER THE SYSTEM DYNAMICS NOISE STRENGTH, QTM.  QTM MUST'),...
  DISP('BE SQUARE WITH DIMENSIONS OF THE NUMBER OF COLUMNS OF GDM.'),COL=S,..
  INQUIRE QTM 'QTM = ',...
  [R,S]=SIZE(QTM);...
  IF R<>COL, DISP('WRONG NUMBER OF ROWS.  TRY AGAIN.'),...
    ELSE IF S<>COL, DISP('WRONG NUMBER OF COLUMNS.  TRY AGAIN.'),...
      ELSE DUMMY=1;...
  END,END,...
 END,...
END
```

```
DISP(' ')
//
// Enter the PTO matrix, the system covariance initial condition.
//
DUMMY=0;
WHILE DUMMY<1,...
 DISP('NOW ENTER THE SYSTEM COVARIANCE INITIAL CONDITION,PTO. PTO MUST'),...
 DISP('BE SQUARE WITH DIMENSIONS OF THE NUMBER OF TRUTH MODEL STATES'),NSTM,.
 INQUIRE PTO 'YOU MAY ENTER A ZERO. PTO = ',...
...//
...// If PTO is 0, make it the right dimensions
...//
 [R,S]=SIZE(PTO); IF R=1, IF S=1,...//Can't do next step if PTO isn't
a scalar
  IF PTO=0,PTO=0*EYE(NSTM);END,...
 END,END,...
...//
...// Ensure PTO has the correct dimensions
...//
 [R,S]=SIZE(PTO);...
 IF R<>NSTM, DISP('WRONG NUMBER OF ROWS.  TRY AGAIN.'),...
   ELSE IF S<>NSTM, DISP('WRONG NUMBER OF COLUMNS.  TRY AGAIN.'),...
      ELSE DUMMY=1;...
 END,END,...
END,END
DISP(' ')
//
// Input the H matrix.  It should have the same number of columns as
//  the number of states, NSTM
//
DUMMY=0;
WHILE DUMMY<1,...
 DISP('NOW ENTER THE H MATRIX. THE H MATRIX IS THE TRUTH MODEL SYSTEM'),...
 DISP('MEASUREMENT MATRIX.  IT MAY BE THE SAME AS THE C MATRIX, IN '),...
 DISP('WHICH CASE JUST ENTER CTM AT THE PROMPT. HTM MUST HAVE THE SAME'),...
 DISP('NUMBER OF COLUMNS AS THE NUMBER OF STATES'),NSTM,...
 INQUIRE HTM 'H MATRIX (HTM) =  ',...
 [R,S]=SIZE(HTM); IF S=NSTM, DUMMY=1;...
   ELSE DISP('WRONG NUMBER OF COLUMNS. TRY AGAIN'),END,...
END
TMNMEAS=R; // TMNMEAS is the Truth Model Number of MEASurements
//
// Enter R of the truth model
//
DUMMY=0;
WHILE DUMMY<1,...
 DISP('NOW ENTER RTM, THE TRUTH MODEL MEASUREMENT NOISE.  RTM MUST BE'),...
 DISP('SQUARE, WITH DIMENSIONS OF THE NUMBER OF HTM ROWS'),ROWS=TMNMEAS,...
 INQUIRE RTM 'YOU MAY ENTER A ZERO.  RTM = ',...
...//
...// If RTM is 0, make it the right dimensions
...//
 [R,S]=SIZE(RTM); IF R=1, IF S=1,...//Can't do next step if PTO isn't
a scalar
  IF RTM=0,RTM=0*EYE(TMNMEAS);END,...
 END,END,...
```

```
...//
...// Ensure RTM has the correct dimensions
...//
 [R,S]=SIZE(RTM);...
 IF R<>TMNMEAS, DISP('WRONG NUMBER OF ROWS.  TRY AGAIN.'),...
   ELSE IF S<>TMNMEAS, DISP('WRONG NUMBER OF COLUMNS.  TRY AGAIN'),...
     ELSE DUMMY=1;...
 END,END,...
END
//
// Form the system matrix STM.
//
STM=[ATM,BTM;CTM,DTM];NSTM=NS;
//
// Now display the truth model to the user.
//
DISP('THE TRUTH MODEL IS (HIT RETURN KEY AFTER EACH DISPLAY):')
ATM,PAUSE,TTM,PAUSE,BTM,PAUSE,CTM,PAUSE,DTM,PAUSE,GTM,PAUSE,HTM,PAUSE,QTM
PAUSE,PTO,PAUSE,RTM,PAUSE
//
// End of TRUMODEL command file
//
CLEAR DUMMY NS M N P Q R S //Clear extraneous variables
CLEAR FLAG ROWS COL
RETURN
```

```
// IMPLICIT.MODEL command file
//
//*******************************************************************
// The IMPLICIT.MODEL command file interactively prompts the user to input
//  the implicit model. The implicit model is of the form:
//     x(Ti+1) = A x(Ti)
//
// Variable Definitions:  AIM = Implicit model A matrix
//                       NSCM = Number of states of command model
//
// This command file is called by the MODELINPUTS and PI command files.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (12 Jan)
//*******************************************************************
//
DISP('THE DESIGN MODEL MUST BE INPUT BEFORE INPUTTING THE IMPLICIT MODEL.')
DISP('THE IMPLICIT MODEL AFFECTS THE SYSTEM GAINS, MAKING THE CONTROLLED')
DISP('SYSTEM MORE ROBUST IN THE FACE OF UNCERTAINTIES OR FAILURES.  THE')
DISP('IMPLICIT MODEL (AIM) CONTAINS DESIRED STABILITY QUALITIES YOU WOULD')
DISP('WANT THE PLANT TO POSSESS.  ONE APPROACH IS TO TRY TO MAKE THE')
DISP('EIGENVECTORS OF THE AIM MATRIX ORTHOGONAL. THE EIGENVALUES, OF')
DISP('COURSE, SHOULD BE NEGATIVE.')
PAUSE
//
// For implict model following in the PI controller to work, the number
//  of rows of AIM must equal those of CDM.
//
```

```
DISP('NOW ENTER THE IMPLICIT MODEL.')
DUMMY=0;
WHILE DUMMY<1,...
 DISP('THE IMPLICIT MODEL MUST BE SQUARE WITH DIMENSIONS OF THE NUMBER
OF'),...
 DISP('DESIGN MODEL INPUTS'),NINPUTS,...
 INQUIRE AIM 'IMPLICIT MODEL (AIM)=',...
 [NS,N]=SIZE(AIM); IF NS<>NINPUTS,...
   DISP('WRONG NUMBER OF ROWS.  TRY AGAIN.'),...
  ELSE IF N<>NINPUTS, DISP('WRONG NUMBER OF COLUMNS.  TRY AGAIN.'),...
     ELSE DUMMY=1;
 END,END,...
END
//
// Show the user the AIM eigenvalues and eigenvectors
//
DISP('THE IMPLICIT MODEL AND ITS EIGENVALUES AND EIGENVECTORS ARE')
AIM, PAUSE
[EVEC,EVAL]=EIG(AIM);
EVEC,PAUSE,EVAL=DIAG(EVAL)
PAUSE
CLEAR N NS EVEC EVAL DUMMY
//
// End of IMPLICIT.MODEL command file
//
RETURN
```

```
//SAVEFILE Command file
//
//****************************************************************************
//
// This command file allows the user to input a file name as a text string.
//  This command file is called by the MODELINPUTS and OVERALL
//  command files.  The reason it is a seperate command file is because
//  MATRIXX threw up on me when I tried including these commands in Option
//  1 of the MODELINPUTS command file.  As a seperate command file, however,
//  these commands work.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M.
// Version 1.0  (6 Jan)
//****************************************************************************
//
DISP('TYPE THE FILENAME WITH SINGLE QUOTATION MARKS AROUND THE FILENAME.')
INQUIRE FILENAME 'FILENAME IS '
SAVE (FILENAME)
CLEAR FILENAME
//
// End of SAVEFILE command file
//
RETURN
```

B.3 CGTPI and Associated Sub-file

```
// CGTPI Command file
//
//***********************************************************************
//
// This command file allows for the generation of the deterministic
//  part of an optimal stochastic controller. This deterministic
//  part generates a command generator tracker with proportional plus
//  integral feedback.
//
// This function is called by the CGTPIKF command file. It has the option
//  to call the PI and CGTPI.ANALYZE command files and the CGT function.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (19 Jan)
//***********************************************************************
//
// Initialize the CGT and BARRAUD functions and create the PIMAT matrix.
//
DEFINE 'CGT.';
DEFINE 'BARRAUD.';
ERASE // Clear the screen
SDMD=DISCRETIZE(SDM,NSDM,DT);
[ADMD,BDMD,CDM,DDM]=SPLIT(SDMD,NSDM);// The C and D matrices do not change
PIMATINV=[ADMD-EYE(NSDM),BDMD;CDM,DDM];
D=DET(PIMATINV); IF D<1.0D-15, PIMAT=PINV(PIMATINV);...
  DISP('A PSEUDOINVERSE WAS USED TO GENERATE THE PIMAT MATRIX'),PAUSE,...
  ELSE PIMAT=INV(PIMATINV);END
```

```
CLEAR D PIMATINV
//
// Discretize the command model
//
SCMD=DISCRETIZE(SCM,NSCM,DT);
[ACMD,BCMD,CCM,DCM]=SPLIT(SCMD,NSCM);// The C and D matrices don't change
X=0; // Initialized for later use (it has to exist before it's CLEARed)
//
// Define a menu for the appropriate functions.
//
DETMENU = ['    CGTPI MENU    '
           '   OPEN LOOP CGT  '
           '   PI REGULATOR   '
           'CLOSED LOOP CGT/PI'
           '  ANALYZE CGT/PI  '
           '  SAVE VARIABLES  '
           'LOOK AT VARIABLES '
           '       QUIT       '];
//
// Now use the menu.
//
OPTION=0;
WHILE OPTION<>7,...
 OPTION=MENU(DETMENU);...
 IF OPTION=1,... // Open loop CGT
  [EVAL]=EIG(ADM);...// If the system is unstable, don't let the user
continue
  CGTFLAG=0;...
 FOR DUMMY=1:NSDM,...
  ZEVAL=REAL(EVAL(DUMMY));...
  IF ZEVAL>0, CGTFLAG=1;END,...
 END,...
...//
...// Save excess variables to prevent MATRIXX from exceeding variable
...//  limits.
...//
  IF CGTFLAG=0,...
   FSAVE 'GARBAGE',...
   CLEAR,...
   LOAD 'GARBAGE' PIMAT SCM NSCM ANM EX EY DT,...
   [A11P,A21P,A22P,A13P,A23P]=CGT(PIMAT,SCM,NSCM,ANM,EX,EY,DT);...
   LOAD 'GARBAGE',...
   A11=A11P;A21=A21P;A22=A22P;A13=A13P;A23=A23P;...
   CLEAR A11P A21P A22P A13P A23P,...
   KX=0*ONES(NINPUTS,NSDM);KZ=0*EYE(NINPUTS);... \\ for later use
   A21,A22,PAUSE,A23,PAUSE,...
  ELSE DISP('SYSTEM UNSTABLE. PURSUE CLOSED LOOP CGT.'),PAUSE,...
  END,...
  CLEAR CGTFLAG DUMMY ZEVAL EVAL,...
 END,...
...//
 IF OPTION=2,... // Generate PI part of controller only
  EXECUTE ('PI.');...//CHECK OUT FORM OF STAIGHT PI CONTROLLER
  KXM=KX,KX,KZ,PAUSE,... //Display kx and kz to the user
 END...
 IF OPTION=3,... // Generate closed loop CGT/PI control law
```

```
    EXECUTE ('PI.');... // PI uses OPTION as a flag
    KXM=KX*A11+A21 KXN=KX*A13+A23;... // From eq. 14-267, Maybeck Vol III
    PAUSE,KX,KZ,PAUSE,...//Display kx and kz to the user
  END,...
  IF OPTION=4,... // Analyze the CGT/PI controlled system
    EXECUTE('CGTPI.ANALYZE'),...
  END,...
  IF OPTION=5,... // Save all variables on the stack
    DISP('YOU WILL NOW SAVE ALL VARIABLES PRESENTLY ON THE STACK TO A FILE.'),
    EXECUTE('SAVEFILE.'),...
  END,...
  IF OPTION=6, INQUIRE X 'ENTER VARIABLE NAME YOU WISH TO SEE (X):  ',X,PAUSE,
   CLEAR X ,...
  END,...
END
CLEAR DETMENU OPTION X
//
// End of CGTPI command file
//
RETURN
```

B.3.1 CGT

```
//[A11,A21,A22,A13,A23]=CGT(PIMAT,SCM,NSCM,ANM,EX,EY,DT)
//
//*************************************************************************
//
// This function generates the gain matrix (A11-A23) for gain generation
//
// The CGT function is called by the PI command file.  It calls the
//  BARRAUD function.  Inputs to this function are the PI matrix,
//  which was formulated in the PI command file, the command model
//  system matrix and number of states, the noise model A matrix (and
//  EX and EY associated with the noise model), and the sample time DT.
//
//  This CGT function was written by Capt. Steve Payson, 1988.
//  Version 1.0
//
//************************************************************************
//
//  Discretize the matrices and split the system matrix into its components
//
SCMD=DISCRETIZE(SCM,NSCM,DT);
[ACMD,BCMD,CCMD,DCMD]=SPLIT(SCMD,NSCM);
//
// Noise models will not be used in this version of the software, but
//  later versions may incorporate them, so the equations incorporate
//  discretized versions of these matrices, which for now I'll set
//  equal to zero.
EXD=0*EX;EYD=0*EY;ANMD=0*ANM;
```

```
CLEAR EX EY ANM SCM NSCM DT SCMD //T o reduce the chance of name overflow
//
// EX HAS NSDM(DESIGN MODEL) ROWS, PIMAT11 HAS DIMENSION OF A(DESIGN MODEL)
//
[M,N]=SIZE(EXD);// M=NSDM
[R,C]=SIZE(PIMAT);
PIMAT11=PIMAT(1:M,1:M);
PIMAT22=PIMAT(M+1:R,M+1:R);
PIMAT12=PIMAT(1:M,M+1:R);
PIMAT21=PIMAT(M+1:R,1:M);
[M,N]=SIZE(ACMD);
DUMMY1=ACMD-EYE(M); DUMMY2=-PIMAT12*CCMD;
//
// BARRAUD function solves for X in  AXB-X=Q
//
A11=BARRAUD(PIMAT11,DUMMY1,DUMMY2);
A21=PIMAT21*A11*DUMMY1 + PIMAT22*CCMD;
A22=PIMAT21*A11*BCMD + PIMAT22*DCMD;
//
[M,N]=SIZE(ANMD);
DUMMY1=ANMD-EYE(M); DUMMY2=PIMAT11*EXD + PIMAT12*EYD;
A13=BARRAUD(PIMAT11,DUMMY1,DUMMY2);
A23=PIMAT21*A13*DUMMY1 - PIMAT21*EXD - PIMAT22*EYD;
//
// End CGT function
//
RETF
```

```
//[X]=BARRAUD(A,B,Q)
//
// This function solves for X in  AXB - X = Q.
// It is based on Barraud's algorithm as outlined in IEEE Tran.
// Automatic Control, AC-22, p 883-885, 1977
//
// Syntax check
EIGA=EIG(A); EIGB=EIG(B);
[M,N]=SIZE(EIGA);
[P,R]=SIZE(EIGB);
FOR I=1:M,FOR J=1:P,...
  EIGAB=EIGA(I)*EIGB(J);...
  IF EIGAB=1,DISP('NO SOLUTION TO A11 OR A13'),RETF,END,...
END,END
//
// End syntax check
//
[M,N]=SIZE(A); //A,B,and Q are square, by def.
[P,R]=SIZE(B);
[UA,ASTAR]=HESSENBERG(A);
IF M<3,ASTAR=A;UA=EYE(M);
[VA,APRIME]=SCHUR(ASTAR);
WA=UA*VA;
[UB,BSTAR]=HESSENBERG(B);
IF P<3,BSTAR=B;UB=EYE(P);
```

```
[VB,BPRIME]=SCHUR(BSTAR);
WB=UB*VB;
QPRIME=WA'*Q*WB;
CLEAR EIGA EIGB UA VA UB VB ASTAR BSTAR //Clear extraneous variables
//
// The following solves for each element of Xprime.
//
FOR COUNTER=1:M, K=M-COUNTER+1; FOR L=1:P,...
...//
...// Compute S term.
...//
     IF L=1,S=0; ELSE ... // That is, S=0 for the first column of A'X'B'
      S=0;...
      FOR J=1:L-1,...
       S=S+APRIME(K,K)*XPRIME(K,J)*BPRIME(J,L);...
     END,END,...
...//
...// Compute R term.
...//
     IF K=M,R=0; ELSE ... // That is, R=0 for last column of A'X'B'
      R=0;...
      FOR I=K+1:M, FOR J=1:L,...
       R=R+APRIME(K,I)*XPRIME(I,J)*BPRIME(J,L);...
     END,END,END,...
...//
...// Compute C and XPRIME
...//
     C(K,L)=QPRIME(K,L)-R-S;...
```

```
    Z=APRIME(K,K)*BPRIME(L,L);...
    IF Z=1, DISP('No solution to A11 or A13'),RETF,...
    ELSE  XPRIME(K,L)=C(K,L)/(Z-1);END,...
...//
END,END
//
//
//Compute X from XPRIME, and check if it is correct
//
X=WA*XPRIME*WB';
D=ABS(A*X*B-X-Q);E=.01*ONES(M,P);
FOR I=1:M, FOR J=1:P,...
IF D(I,J)>E(I,J), DISP('BARRAUD DID NOT WORK CORRECTLY'),...
END,END
RETF
```

B.3.2 PI

```
// PI Command File
//
//*********************************************************************
//
// This command file generates the PI of CGT/PI control law. It uses
//  the variable OPTION from the CGTPI command file as a flag. If
//  OPTION is equal to 3, a closed loop CGT/PI control law is pursued.
//  Otherwise, just the PI controller is pursued.
//
// Variable Definitions:  M = dimension of design model (NSDM)
//                        J = number of design model inputs
//                     AAUG = augmented A matrix, used for cost function
//                     BAUG = augmented B matrix, used for cost function
//                     CAUG = augmented C matrix, used for cost function
//
//
// This command file is called by the CGTPI command file.
//  It has the option to call the CGT function.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (27 Jan)
//**********************************************************************
//
// Initialize the necessary functions
//
DEFINE 'EXPLICIT.'
DEFINE 'IMPLICIT.'
DEFINE 'DISCOST.'
ERASE // Clear the screen
//
// Form augmented matrices for cost generation (with DISCOST function).
//
[M,NSDM]=SIZE(ADM); // M=NSDM, because ADM is square
[M,J]=SIZE(BDM); // Number rows of BDM = number rows of ADM
AAUG=[ADM,BDM;0*ONES(J,NSDM),0*EYE(J)]; //EQ 14-216, P141,VOL III
BAUGD=[0*ONES(M,J);EYE(J)]; // Same equation
BAUG=[0*ONES(M,J);EYE(J)]; // Same equation
CAUG=[CDM,-1*CCM];
//
// AAUG has M+J rows and columns
// BAUG has M+J rows, J columns
// CDM has M columns and J rows, CCM has J rows and columns
// CAUG has J rows, M+J columns
//
DISP('PURSUE EXPLICIT MEANS TO PURSUE A STANDARD FORM PI CONTROLLER')
DISP('WHICH WEIGHTS PERTURBATION DEVIATIONS FROM ZERO. IMPLICIT')
DISP('MODEL FOLLOWING WEIGHTS DEVIATIONS FROM A MODEL WITH DESIRED')
DISP('STABILITY ROBUSTNESS CHARACTERISTICS.  PURSUING BOTH SIMPLY')
DISP('ADDS THE TWO COST WEIGHTING MATRICES TOGETHER.')
PAUSE
//
// Define a menu for the appropriate functions
//
COSTMENU = ['      PI MENU        '
            'INPUT COST MATRICES '
```

```
              'INPUT IMPLICIT MODEL'
              '  PURSUE EXPLICIT   '
              '  PURSUE IMPLICIT   '
              '    PURSUE BOTH     '
              ' ALTER XIE ELEMENT  '
              ' LOOK AT VARIABLES  '
              'CONTINUE WITH DESIGN'];
//
// Now use the menu (Z is the option parameter)
//
Z=0;
WHILE Z<>8,...
 Z=MENU(COSTMENU); CHECK=0;... // CHECK is a flag variable used later
 IF Z=1,... // Input cost matrices (Y,UM,QI,RI, and UR)
  EXECUTE ('INPUTCOST.');...
 END,...
...//
 IF Z=2, EXECUTE('IMPLICIT.MODEL'),END,...
...//
 IF Z=3,... // Pursue explicit path only
  [XC11,XC12,XC22]=EXPLICIT(Y,UM,CDM,DDM);...
  XIE=[XC11,XC12;XC12',XC22],PAUSE,...
  QIHAT=0;RIHAT=0;SIHAT=0;...//These must exist for FSAVE command later
on
...//
...// Test to ensure XIE is positive semidefinite
...//
  CHECK=0;...
  EVAL=EIG(XIE);...
  [P,Q]=SIZE(XIE);...// P=Q by this point
  FOR ZZ=1:P,...
    XIEEVAL=EVAL(ZZ,1);...
    IF XIEEVAL<0, CHECK=1; END,...
  END,...
  IF CHECK=1, DISP('YOU ENDED UP WITH A WEIGHTING MATRIX THAT IS NOT'),...
     DISP('POSITIVE SEMIDEFINITE. FIX THIS BEFORE CONTINUING'),PAUSE,END,...
 END,...
...//
 IF Z=4,... // Pursue implicit path only
  [QIHAT,SIHAT,RIHAT]=IMPLICIT(QI,RI,CDM,ADM,BDM,AIM);...
  XIE=[QIHAT,SIHAT;SIHAT',RIHAT],PAUSE,...
  XC11=0;XC12=0;XC22=0;...//These must exist for FSAVE command later on
...//
...// Test to ensure XIE is positive semidefinite
...//
  CHECK=0;...
  EVAL=EIG(XIE);...
  [P,Q]=SIZE(XIE);...// P=Q by this point
  FOR ZZ=1:P,...
    XIEEVAL=EVAL(ZZ,1);...
    IF XIEEVAL<0, CHECK=1; END,...
  END,...
  IF CHECK=1, DISP('YOU ENDED UP WITH A WEIGHTING MATRIX THAT IS NOT'),...
     DISP('POSITIVE SEMIDEFINITE. FIX THIS BEFORE CONTINUING'),PAUSE,END,...
 END,...
...//
```

```
 IF Z=5,... // Pursue combined implicit and explicit path
  [XC11,XC12,XC22]=EXPLICIT(Y,UM,CDM,DDM);...
  [QIHAT,SIHAT,RIHAT]=IMPLICIT(QI,RI,CDM,ADM,BDM,AIM);...
  XIE=[(XC11+QIHAT),(XC12+SIHAT);(XC12'+SIHAT'),(XC22+RIHAT)],PAUSE,...
...//
...// Test to ensure XIE is positive semidefinite
...//
  CHECK=0;...
  EVAL=EIG(XIE);...
  [P,Q]=SIZE(XIE);...// P=Q by this point
  FOR ZZ=1:P,...
    XIEEVAL=EVAL(ZZ,1);...
    IF XIEEVAL<0, CHECK=1;END,...
  END,...
  IF CHECK=1, DISP('YOU ENDED UP WITH A WEIGHTING MATRIX THAT IS NOT'),...
     DISP('POSITIVE SEMIDEFINITE. FIX THIS BEFORE CONTINUING'),PAUSE,END,...
 END,...
...//
 IF Z=6, XIE,...
  INQUIRE P 'ENTER ROW ELEMENT OF XIE YOU WISH TO CHANGE : ',...
  INQUIRE Q 'ENTER COLUMN ELEMENT OF XIE YOU WISH TO CHANGE : ',...
  DISP('NOW ENTER THE DESIRED VALUE.  SYMMETRY WILL BE MAINTAINED.'),...
  DISP('YOU WILL BE TOLD IF XIE IS NOT POSITIVE SEMI-DEFINITE.'),...
  INQUIRE ZZ 'ENTER VALUE:   ',...
  XIE(Q,P)=ZZ;XIE(P,Q)=ZZ;...
...//
...// Test to ensure XIE is positive semidefinite
...//
```

```
  CHECK=0;...
  EVAL=EIG(XIE);...
  [P,Q]=SIZE(XIE);...// P=Q by this point
  FOR ZZ=1:P,...
    XIEEVAL=EVAL(ZZ,1);...
    IF XIEEVAL<0, CHECK=1;END,...
  END,...
  IF CHECK=1, DISP('YOU ENDED UP WITH A WEIGHTING MATRIX THAT IS NOT'),...
     DISP('POSITIVE SEMIDEFINITE. FIX THIS BEFORE CONTINUING'),PAUSE,END,...
 END,...
...//
 IF Z=7,INQUIRE X 'ENTER VARIABLE YOU WANT TO SEE (X): ',X,PAUSE,...
 CLEAR X ,...
 END,...
...//
END
//
// XIE is square with dimension p=m+j, UR is square with dimension j.
//  The program now continues, and allows the user to make changes
//  in XIE.
//
Z=0;  [P,Q]=SIZE(XIE);// P is regenerated in case a user skips options
1-4
WHILE Z<P,...//P is the number of rows of XIE
  DISP('XIE IS THE WEIGHTING MATRIX FOR THE AUGMENTED STATE EQ [X,U]'),...
  XIE,...
  DISP('YOU MAY NOW ALTER THIS MATRIX IF YOU WISH. OTHERWISE, TYPE XIE'),...
  INQUIRE XIE 'XIE =',...
```

```
...//
...// Test to ensure XIE is positive semidefinite
...//
  CHECK=0;...
  EVAL=EIG(XIE);...
  [P,Q]=SIZE(XIE);...// Recheck the size of XIE, since the user changed
it
  FOR ZZ=1:P,...
    XIEEVAL=EVAL(ZZ,1);...
    IF XIEEVAL<0, DISP('YOU ENDED UP WITH A WEIGHTING MATRIX THAT IS NOT'),..
     DISP('POSITIVE SEMIDEFINITE. FIX THIS BEFORE CONTINUING'),XIE,PAUSE,Z=0;
    ELSE Z=Z+1;...// When all eigenvalues have passed this test, z=p
  END,...
END
CLEAR Z ZZ EVAL XIEEVAL COSTMENU CHECK// Clear extraneous variables
//
// Allow the user to specify SXU, which will be of dimension p x j.
//
DISP('YOU MAY NOW SPECIFY OFF DIAGONAL TERMS. IF YOU DO NOT WANT TO')
DISP('PURSUE THIS, TYPE A ZERO. OTHERWISE THE DIMENSIONS MUST BE')
DISP('CONSISTENT WITH XIE AND UR, THAT IS, HAVE THE FOLLOWING ROWS AND
COLUMNS')
DUMMY=0;
WHILE DUMMY<1,...
 ROWS=P,COL=J,... //This will display the number of rows and columns to
the user
 ...                // XIE is of dimension P, UR is of dimension J
 INQUIRE SXU 'SXU MATRIX=',...
```

```
...//
...// If user inputs a zero, make SXU the correct dimensions
...//
 [ROWS,Q]=SIZE(SXU);IF ROWS=1,IF Q=1,...//Can't do next step if SXU not
a scalar
     IF SXU=0,SXU=0*ONES(P,J), END,END,END,...
...//
...// Ensure SXU has the correct dimensions. If a 0 was input, it already
...//  will, but we'll check anyway.
...//
 [ROWS,Q]=SIZE(SXU);...
 IF ROWS<>P, DISP('THE NUMBER OF ROWS WAS WRONG. TRY AGAIN'),...
   ELSE IF Q=J, DUMMY=1; ELSE...
     DISP('THE NUMBER OF COLUMNS WAS WRONG. TRY AGAIN'),...
 END,END,...
END
CLEAR DUMMY ROWS COL // Clear extraneous variables
DISP('THIS WILL TAKE A WHILE.  PLEASE BE PATIENT.')
//
// Now find the gains GCBAR
// EVAL is the closed loop eigenvalues, GCA is the optimal state feedback
//  gain matrix.
//
[AAUGD,BAUGD1,XIED,URD,SXUD]=DISCOST(AAUG,BAUG,DT,XIE,UR,SXU);...
[EVAL,GCA]=DREGULATOR(AAUGD,BAUGD,XIED,URD,SXUD);...//MATRIXX function
//
//SXUD' is required by the DREGULATOR command for dimensionality reasons,
// rather than SXUD. GCA has p (m+j) rows
```

```
//
GC1=GCA(:,1:M);...// All rows, columns 1 through M (M=# states of design
model)
GC2=GCA(:,M+1:P);...// All rows, columns M+1 through M+R
//
// Divide PIMAT into its component matrices (M is the number of design
model
//  states). PIMAT is square with dimension p. PIMAT was formed in the
CGT
//  command file, which calls this PI command file.
//
PIMAT11=PIMAT(1:M,1:M);
PIMAT22=PIMAT(M+1:P,M+1:P);
PIMAT12=PIMAT(1:M,M+1:P);
PIMAT21=PIMAT(M+1:P,1:M);
//
KX=GC1*PIMAT11 + GC2*PIMAT21; // Equation 14-277a, p145, Maybeck Vol III
KZ=GC1*PIMAT12 + GC2*PIMAT22; // Same equation
//
IF OPTION=3,... // This is the flag for the closed loop CGT/PI controller
...//
...// Save excess variables to prevent MATRIXX from exceeding variable
...//  limits (which can happen with the extent levels generated during
...//  the CGT function).
...//
  FSAVE 'GARBAGE',...// A dummy file which the user can later delete.
  CLEAR,...
  LOAD 'GARBAGE' PIMAT SCM NSCM ANM EX EY DT ,...
  [A11P,A21P,A22P,A13P,A23P]=CGT(PIMAT,SCM,NSCM,ANM,EX,EY,DT);...
  LOAD 'GARBAGE',...
  A11=A11P;A21=A21P;A13=A13P;A22=A22P;A23=A23P;...
  CLEAR A11P A21P A13P A22P A23P ,...// All dummy variables
 ELSE A11=0;A21=0;A22=0;A13=0;A23=0;...
END
CLEAR PIMAT11 PIMAT12 PIMAT21 PIMAT22 R C M GCA P Q
CLEAR AAUG AAUGD BAUG BAUGD BAUGD1 CAUG X GC1 GC2
//
// End of PI command file
//
RETURN
```

```
// INPUTCOST Command File
//
//*********************************************************************
//
// This command file allows for inputting the basic cost weighting
//  matrices.
//
// Variable Definitions:   Y = cost on output deviations
//                        UM = cost on control magnitudes
//                        UR = cost on control rates
//                         J = number of inputs
//                        QI = cost on deviations from model output
//                        RI = cost on deviations from model control magnitud
//
// Constaints on the cost matrices are that they all must have
//  dimensions jxj (j=number of inputs). UR must be positive definite,
// all other matrices must be positive semidefinite.
//
// This function is called by the PI command file. 'j' is the only
//  variable it requires for operation. All other variables are
//  requested internal to this function.
//
// This function was written by Captain Steve Payson, 1988
// Version 1.0  (8 Jan)
//*********************************************************************
//
// Allow the user to load the costs from a file
//
```

```
DISP('IF THE COSTS YOU WISH TO INPUT ARE IN A FILE THAT YOU WANT TO')
DISP('LOAD, ENTER A 0 <ZERO>. OTHERWISE, TYPE A 1.  BE CAREFUL HERE,')
DISP('AS ANY VARIABLES IN THE FILE BEING READ WILL OVERWRITE DATA')
DISP('PRESENTLY BEING USED BY THIS PROGRAM.')
INQUIRE DUMMY 'ENTER OPTION: 0  OR 1     '
IF DUMMY=0, EXECUTE('INPUTFILE.'),...
  DISP('EVEN THOUGH YOU LOADED THE COSTS, YOU STILL HAVE TO GO THROUGH'),...
  DISP('THE FOLLOWING STEPS TO ENSURE PROPER SYNTAX OF YOUR COSTS.'),...
END
//
// Proceed with the rest of this program
//
DISP('YOU WILL NOW INPUT THE COST WEIGHTING MATRICES. YOU MUST')
DISP('INPUT SOMETHING FOR ALL PROMPTS. IF A MATRIX HAS BEEN DEFINED')
DISP('PREVIOUSLY AND YOU DO NOT WISH TO CHANGE IT, RETYPE THE MATRIX NAME.')
DISP('MATRICES MUST BE POSITIVE SEMIDEFINITE, AND HAVE THE SAME NUMBER')
DISP('OF ROWS AND COLUMNS AS THE NUMBER OF INPUTS, WHICH ARE:'), NINPUTS
DISP('HIT THE RETURN KEY TO PROCEED')
J=NINPUTS;PAUSE
//
//
DISP('EXPLICIT COST MATRICES: Y AND UM')
DISP('THE Y MATRIX WEIGHTS OUTPUT DEVIATIONS')
DISP('THE UM MATRIX WEIGHTS CONTROL MAGNITUDES')
//
DUMMY=0; // DUMMY is used as a dummy variable
WHILE DUMMY<1,...
 INQUIRE Y 'Y MATRIX =',...
 [P,Q]=SIZE(Y); IF P<>J, DISP('WRONG NUMBER OF ROWS. TRY AGAIN'),...
   ELSE IF Q<>J, DISP('WRONG NUMBER OF COLUMNS. TRY AGAIN'),...
     ELSE DUMMY=1;EVAL=EIG(Y);...//test for semi-definiteness
     FOR ZZ=1:P,...
       ZEVAL=EVAL(ZZ,1);...
       IF ZEVAL<0, DISP('MATRIX MUST BE POSITIVE SEMIDEFINITE'),...
         DUMMY=-P;END,...
   END,END,END,...
END
//
WHILE DUMMY<2,...
 INQUIRE UM 'UM MATRIX =',...
 [P,Q]=SIZE(UM); IF P<>J, DISP('WRONG NUMBER OF ROWS. TRY AGAIN'),...
   ELSE IF Q<>J, DISP('WRONG NUMBER OF COLUMNS. TRY AGAIN'),...
     ELSE DUMMY=2;EVAL=EIG(UM);...//test for semi-definiteness
     FOR ZZ=1:P,...
       ZEVAL=EVAL(ZZ,1);...
       IF ZEVAL<0, DISP('MATRIX MUST BE POSITIVE SEMIDEFINITE'),...
         DUMMY=-P;END,...
   END,END,END,...
END
//
//
DISP('IMPLICIT COST MATRICES: QI AND RI')
DISP('THE QI MATRIX WEIGHTS DEVIATIONS FROM THE COMMAND MODEL OUTPUTS')
DISP('THE RI MATRIX WEIGHTS DEVIATIONS FROM THE COMMAND MODEL INPUTS')
//
WHILE DUMMY<3,...
```

```
 INQUIRE QI 'QI MATRIX =',...
 [P,Q]=SIZE(QI); IF P<>J, DISP('WRONG NUMBER OF ROWS. TRY AGAIN'),...
   ELSE IF Q<>J, DISP('WRONG NUMBER OF COLUMNS. TRY AGAIN'),...
     ELSE DUMMY=3;EVAL=EIG(QI);...//test for semi-definiteness
     FOR ZZ=1:P,...
       ZEVAL=EVAL(ZZ,1);...
       IF ZEVAL<0, DISP('MATRIX MUST BE POSITIVE SEMIDEFINITE'),...
         DUMMY=-P;END,...
   END,END,END,...
END
//
WHILE DUMMY<4,...
 INQUIRE RI 'RI MATRIX =',...
 [P,Q]=SIZE(RI); IF P<>J, DISP('WRONG NUMBER OF ROWS. TRY AGAIN'),...
   ELSE IF Q<>J, DISP('WRONG NUMBER OF COLUMNS. TRY AGAIN'),...
     ELSE DUMMY=4;EVAL=EIG(RI);...//test for semi-definiteness
     FOR ZZ=1:P,...
       ZEVAL=EVAL(ZZ,1);...
       IF ZEVAL<0, DISP('MATRIX MUST BE POSITIVE SEMIDEFINITE'),...
         DUMMY=-P;END,...
   END,END,END,...
END
//
//
DISP('THE UR MATRIX WEIGHTS CONTROL RATES AND IS USED IN BOTH EXPLICIT')
DISP('AND IMPLICIT MODES. UR MUST BE POSITIVE DEFINITE.')
WHILE DUMMY<5,...
 INQUIRE UR 'UR MATRIX =',...
 [P,Q]=SIZE(UR); IF P<>J, DISP('WRONG NUMBER OF ROWS. TRY AGAIN'),...
   ELSE IF Q<>J, DISP('WRONG NUMBER OF COLUMNS. TRY AGAIN'),...
     ELSE DUMMY=5;EVAL=EIG(UR);...//test for definiteness
     FOR ZZ=1:P,...
       ZEVAL=EVAL(ZZ,1);...
       IF ZEVAL<=0, DISP('MATRIX MUST BE POSITIVE DEFINITE'),...
         DUMMY=-P;END,...
   END,END,END,...
END
Y,UM,PAUSE,QI,RI,PAUSE,UR,PAUSE
CLEAR ZZ ZEVAL EVAL //Clear extraneous variables
//
// End of INPUTCOST command file
//
RETURN




//[XC11,XC12,XC22]=EXPLICIT(Y,UM,CDM,DDM)
//
//***********************************************************************
//
// The EXPLICIT function transforms the Y and UM cost weighting matrices
//  (which weight output deviations and control magnitudes, respectively)
//  into XC11,XC12, and XC22, the explicit cost weighting matrices.
//
// Variable Definitions:    CDM = design model C matrix
//                                  DDM = design model D matrix
//                                    Y = output deviation weighting matrix
//                                   UM = control magnitude weighting matrix
//                                 XC11 = explicit cost matrix = CAUG'*Y*CAUG
//                                 XC22 = explicit cost matrix = UM+DDM'*Y*DDM
//                                 XC12 = explicit cost matrix = CAUG'*Y*DDM
//
// The dimensionality of Y and XC22 is j x j, the number of inputs.
//  XC11 has dimensionality m x m, where m is the number of design
//  model states. XC12 has dimensions m x j.
//
// This function is called by the PI command file.
//
// This function was written by Captain Steve Payson, 1988.
// Version 1.0  (18 Oct)
//****************************************************************************
//
XC11 = CDM'*Y*CDM;
XC22 = UM + DDM'*Y*DDM;
XC12 = CDM'*Y*DDM;
//
// End EXPLICIT function
//
RETF



//[QIHAT,SIHAT,RIHAT]=IMPLICIT(QI,RI,C,A,B,AIM)
```

```
//
//*******************************************************************
//
// The IMPLICIT function transforms the QI and RI weighting matrices
//  (which force the system outputs to mimic the dynamics of a model
//  system) into QIHAT, SIHAT, and RIHAT. QI and RI are input in
//  the PI command file. (SEE MILLER, PAGE A-13)
//
// Variable Definitions:      A = system A matrix
//                            B = system B matrix
//                            C = system C matrix
//                          AIM = implicit model A matrix
//                           QI = weighting matrix, ouputs mimic model
//                           RI = weighting matrix, controls mimic model
//                        QIHAT = implicit cost
//                        SIHAT = implicit cost
//                        RIHAT = implicit cost
//
// When this function is used to generate costs for a CGT/PI controller,
//  then the A, B, and C matrices of this function are the respective
matrices
//  of the design model, and QI, RI and RIHAT are all of dimension j x
j, where
//  j is the number of inputs. QIHAT is m x m, where m is the number of
//  design model states. SIHAT is of dimension m x j.
//
// For implicit model following to work, the number of states (rows and
columns)
```

```
//  of AIM must equal the number of rows of the C matrix.
//
// This function is called by the PI command file.
//
// This function was written by Captain Steve Payson, 1988.
// Version 1.0  (12 Jan)
//*********************************************************************
//
// Check to ensure AIM and C have the correct dimensions
//
[P,Q]=SIZE(AIM);  [R,S]=SIZE(C);
IF P<>R, DISP('THE NUMBER OF STATES OF AIM ARE INCOMPATIBLE FOR IMPLICIT'),..
  DISP('MODEL FOLLOWING TO WORK.'),...
  QIHAT=0;SIHAT=0;RIHAT=0;RETF,END
//
// If AIM and C are the correct dimensions, form the implicit cost weighting
//  matrices.
//
DUMMY = C*A - AIM*C;
QIHAT = DUMMY'*QI*DUMMY;
SIHAT = DUMMY'*QI*C*B;
RIHAT = RI + B'*C'*QI*C*B;
//
// End of Implicit function
//
RETF
```

```
//[Ad,Bd,Rxxd,Ruud,Rxud]=DISCOST(A,B,DT,Rxx,Ruu,Rxu)
//
//**********************************************************************
//
// The DISCOST function was taken from the April 1988 issue of
//  'Ask Dr. Control'.
//
//  DISCOST provides the discrete cost matrices and plant and control
//   matrices associated with their continuous counterparts.
//  DISCOST is called by the PI command file.
//
//**********************************************************************
//
[NS,NC]=SIZE(B);  // NS is the number of states,
// NC is the number of inputs
//
// Syntax checking: all arguments checked for consistency.
//
[M,N]=SIZE('A',1); // A must be NS x NS
IF M=N, IF M<>NS,...
  DISP('Dimensionality of models is wrong'), RETF, END,...
  ELSE, DISP('Dimensionality of models is wrong'), RETF
IF SUM(SIZE('DT',1))<>2,DISP('DT must be a scalar'),RETF
[M,N]=SIZE('Rxx',1);
// Rxx must be NS x NS
IF M=N, IF M<>NS,...
```

```
    DISP('Y and Qi rows must equal number of controlled outputs'),...
    RETF,END,...
  ELSE DISP('Y and Qi must be square'),RETF
//
[M,N]=SIZE('Rxu',1);
// Rxu is optional (default is zero)
IF M=0,Rxu=0*ONES(NS,NC); ...
...// must be NS x NC if specified
ELSE IF M=NS, IF N<>NC,...
   DISP('Off diagonal elements wrong dimension'),...
   RETF, END, ELSE,...
   DISP('Off diagonal elements wrong dimension')
//
// Syntax checking is over, start computations
//
T=SQRT(Ruu);
B=B/T;
Sd=EXP([-A',EYE(NS),0*ONES(NS,NS+NC);0*ONES(NS),-A',Rxx,Rxu/T;...
      0*ONES(NS,2*NS),A,B;0*ONES(NC,3*NS+NC) ]*DT);
Ad   = Sd(2*NS+1:3*NS,2*NS+1:3*NS);
Bd   = Sd(2*NS+1:3*NS,3*NS+1:3*NS+NC)*T;
Rxxd = Ad'*Sd(NS+1:2*NS,2*NS+1:3*NS);
Rxud = Ad'*Sd(NS+1:2*NS,3*NS+1:3*NS+NC)*T;
Ruud = B'*Ad'*Sd(1:NS,3*NS+1:3*NS+NC);
Ruud = Ruu*DT + T'*(Ruud+Ruud')*T;
//
// End of DISCOST function
//
```

```
RETF
```

B.3.3 CGTPI.ANALYZE

```
// CGTPI.ANALYZE command file
//
//*************************************************************************
//
// This command file allows the user to analyze the closed-loop CGT/PI
//  control system (the open-loop CGT can also be analyzed). Limitations
//  exist on how long a MATRIXX line can be (4096 characters), so instead
//  of having one menu, the different options are divided into more menus.
//  One menu would effectively turn this entire command file into one
//  line of code, which results in the 'input buffer' overflowing.
//
// This command file calls the PLOTHELP function, which is a subroutine
//  incorporating commands used in plotting options.  PLOTHELP generates
//  the TIME and STRENGTH vectors.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (4 Feb)
//*************************************************************************
//
DEFINE 'PLOTHELP.'; // User defined function
DEFINE 'PLOTLOOP.'; // User defined function
ERASE // Clear the screen
DISP('THIS PART OF THE PROGRAM ALLOWS YOU TO GET TIME RESPONSE PLOTS.')
DISP('THE FIRST MENU ALLOWS YOU TO MODIFY GAINS, AND TO LOAD NEW FILES')
DISP('SUCH AS A NEW TRUTH MODEL.  THE TRUTH MODEL MUST EXIST BEFORE YOU')
DISP('CONTINUE.  ANYTIME YOU MODIFY ANY ASPECT OF THE SYSTEM (EG, GAINS),')
DISP('YOU MUST ENTER OPTION 3, BUILD THE SYSTEM, BEFORE PURSUING ANY ANALYSIS
DISP('THE SECOND (NEXT) MENU ALLOWS YOU TO GET THE TIME RESPONSE PLOTS.')
PAUSE
//
// Define a menu for the appropriate functions
//
ANMENU= ['CGTPI.ANALYZE MENU1'
         ' MODIFY THE GAINS  '
         '   LOAD A FILE     '
         ' BUILD THE SYSTEM  '
         '    NEXT MENU      '];
//
// Now use the menu (Z is the option parameter)
//
Z=0;
WHILE Z<>4,...
 Z=MENU(ANMENU);...
...//
...// Option 1 allows the user to change gain values
...//
 IF Z=1, DISP('OLD KX = '),KX,...
   ZZ=0; FOR ZZ=1,... // ZZ IS THE FLAG
     [I,J]=SIZE(KX);INQUIRE KX 'NEW KX =',...
     [K,L]=SIZE(KX); IF K<>I,DISP('TRY AGAIN'),ZZ=0;...
       ELSE IF L<>J,DISP('TRY AGAIN'),ZZ=0;...
         ELSE KXM=KX*A11+A21,ZZ=1;...
   END,END,END,...
   DISP('OLD KZ = '),KZ,...
   ZZ=0; FOR ZZ=1,... // ZZ IS THE FLAG
```

```
      [I,J]=SIZE(KZ);INQUIRE KZ 'NEW KZ =',...
      [K,L]=SIZE(KZ); IF K<>I,DISP('TRY AGAIN'),ZZ=0;...
        ELSE IF L<>J,DISP('TRY AGAIN'),ZZ=0;...
          ELSE ZZ=1;...
    END,END,END,...
    DISP('ANYTIME YOU MODIFY ANY ASPECT OF THE SYSTEM, YOU MUST '),...
    DISP('ENTER OPTION 3, BUILD THE SYSTEM, BEFORE PURSUING ANY ANALYSIS'),...
    PAUSE,...
  END,...
...//
...// Option 2 allows the user to input a new file, such as a new truth
...//   model to compare the controller against.
...//
  IF Z=2, EXECUTE('INPUTFILE.'),...
    DISP('IF YOU LOADED A NEW FILE THAT YOU WANT TO COMPARE THE CONTROLLER'),.
    DISP('AGAINST, YOU NEED TO PURSUE OPTION 3, BUILD THE SYSTEM, BEFORE'),...
    DISP('PURSUING ANY ANALYSIS.'),PAUSE,...
  END,...
...//
...// Option 3 builds the system.  It calls the command file CONTROL.
...//
  IF Z=3,  EXECUTE('CONTROL.'),END,...
...//
END
//
DISP('NOW YOU CAN GET TIME RESPONSE PLOTS.  DES MOD IS THE UNCONTROLLED
')
DISP('DESIGN MODEL.  COM MOD IS THE COMMAND MODEL.  TRUTH MODEL OUTPUTS')
DISP('ARE THE TRUTH MODELS OUTPUTS WITH THE CONTROLLER IN THE LOOP.')
DISP('ACTUATOR RESPONSE IS THE RESPONSE OF THE ACTUATORS.  IF NO ACTUATOR')
DISP('DYNAMICS EXIST, THIS IS THE OPTIMAL CONTROL GENERATED BY THE')
DISP('CONTROLLER. IT IS ALSO THE INPUT TO THE TRUTH MODEL BEING EVALUATED.')
PAUSE
//
// Start the next menu
//

ANMENU= ['CGTPI.ANALYZE MENU2'
         'PLOT DES MOD OUTPUT'
         'PLOT COM MOD OUTPUT'
         'TRUTH MODEL OUTPUTS'
         'TRUTH MODEL STATES '
         ' LOOK AT VARIABLES '
         ' ACTUATOR RESPONSE '
         '      QUIT         '];

//
// Now use the menu (Z is the option parameter). PL is a flag for plot
//  hardcopies, which affects how the plot prints (slower but prettier).
//
Z=0;PL=0;
WHILE Z<>7,...
 Z=MENU(ANMENU);...
...//
...// Option 1 allows the user to plot the uncontrolled design model response
...//  YDMD.
```

```
...//
 IF Z=1,...
   DISP('ENTER TIME DURATION'),...
   INQUIRE TIME 'TIME IN SECONDS = ',...
   NPTS=TIME/DT + 1;...
   [N,YDMD1]=DSTEP(SDMD,NSDM,NPTS);...
...//
...// MATRIXX generates outputs responses vs each input.  Obtaining a
...//  consolidated output makes more sense.
...//
   YDMD=0*YDMD1;... // Initialize YDMD
   FOR I=1:NINPUTS,...
    FOR J=1:NINPUTS,...
     YDMD(:,I)=YDMD(:,I)+YDMD1(:,J);...
   END,END,...
...//
   TIME=N*DT;...
   [PL]=PLOTLOOP(TIME,YDMD,PL);...
   PAUSE,...
   ERASE,... // clear the screen
 END,...
...//
...// Option 2 allows the user to plot the command model ideal response,
...//  YCMD.
...//
 IF Z=2,...
   DISP('THE PROGRAM WILL NOW GENERATE THE COMMAND MODEL RESPONSE'),...
   DISP('(YCMD) TO A STEP INPUT'),...
   [TIME,UCM]=PLOTHELP(SCMD,NSCM,NINPUTS,DT,1);... // user defined function
   EXECUTE('ANALYZE.YCMD'),...// This calls a SYSTEM BUILD command file
   YCMD=SIM(TIME,UCM);...// This simulates the com model with UCM as the
input
   PL=PLOTLOOP(TIME,YCMD,PL);...
   ERASE,... // Clear the screen
   END,...
...//
...// Option 3 allows the user to view the controlled system's output
response,
...//  YCONT.
...//
 IF Z=3,...
   DISP('THE PROGRAM WILL NOW GENERATE THE CONTROLLED SYSTEM RESPONSE'),...
   DISP('OF THE TRUTH MODEL (YTM) TO A STEP INPUT'),...
   [TIME,UCM]=PLOTHELP(STMD,NSTM,NINPUTS,DT,1);...
   EXECUTE('ANALYZE.YTM'),...// This calls a SYSTEM BUILD command file
   YTM=SIM(TIME,UCM);...// This simulates the system with UCM as the input
   PL=PLOTLOOP(TIME,YTM,PL);...
 END,...
...//
...// Option 4 lets the user see the controlled system's state response,
...//  XCONT.
...//
 IF Z=4,...
   DISP('THE PROGRAM WILL NOW GENERATE THE CONTROLLED TRUTH MODEL'),...
   DISP('STATE RESPONSE (XTM) TO A STEP INPUT'),...
   [TIME,UCM]=PLOTHELP(SDMD,NSDM,NINPUTS,DT,1);...
```

```
    EXECUTE('ANALYZE.XTM'),...// This calls a SYSTEM BUILD command file
    XTM=SIM(TIME,UCM);...// This simulates the system with UCM as the input
    PL=PLOTLOOP(TIME,XTM,PL);...
  END,...
...//
IF Z=5,INQUIRE X 'ENTER VARIABLE YOU WANT TO SEE (X): ',X,PAUSE,...
 CLEAR X ,...
END,...
...//
IF Z=6,...
 DISP('THE PROGRAM WILL NOW PLOT THE ACTUATOR OUTPUTS (UOPTIMAL)'),...
 DISP('THAT RESULTS FROM THE DESIGN EFFORT.'),...
 [TIME,UCM]=PLOTHELP(SDMD,NSDM,NINPUTS,DT,1);...
 EXECUTE('ANALYZE.ACT'),... // This calls a SYSTEM BUILD command file
 ACTUATOR=SIM(TIME,UCM);...
 DISP('ENTER THE TITLE, WITH QUOTATION MARKS AT BEGINNING AND END.'),...
 INQUIRE TITLE 'ENTER TITLE',...
 PLOT(TIME,ACTUATOR,['TITLE/',TITLE]),...
 INQUIRE PLOTTER 'IF YOU WANT A HARDCOPY ENTER 0. OTHERWISE, ENTER 1.',...
 IF PLOTTER=0, PLOTFLAG=1;EXECUTE('PLOT.HARDCOPY'),END,...
 ERASE,...
 CLEAR TITLE
END,...
...//
END
CLEAR ANMENU Z ZZ I R C PL
BUILD,TOP,MAT // for some reason, MATRIXX wants this done.
//
```

```
// End of CGTPI.ANALYZE command file
//
RETURN
```

```
//[TIME,U]=PLOTHELP(S,NS,NINPUTS,DT,FLAG)
//
//**************************************************************************
//
// This function is called by CGTPI.ANALYZE.  The commands in this
//  function are used repeatedly in the plotting options of the
//  second menu, so these commands were broken out seperately to reduce
//  the chance of input buffer overloading (explained in the CGTPI.ANALYZE).
//  This function prompts the user for the time of the simulation and
//  the strength of the step input, or the time and strength of three
//  pulses.
//
// FLAG is a flag variable.  When FLAG=0, the system being analyzed is
//  continuous.  When FLAG=1, the system is discrete.  This flag variable
//  is required because the MATRIXX function DSTEP sometimes has a
//  fixup-overflow (MATRIXX error message) when a continuous-time system
//  matrix is used with it.
//
//
// This function was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (22 Feb)
//**************************************************************************
//
 DISP('ENTER TIME DURATION OF STEP INPUT.'),...
 INQUIRE TIME 'TIME IN SECONDS = ',...
 IF FLAG=0,...  //This loop is just to generate N
   [TIME,Y]=STEP(S,NS,TIME);... // This creates a matrix of time values
   [I,J]=SIZE(TIME);... // Y and J are unimportant.  I is used later.
 ELSE,...
  NPTS=TIME/DT + 1;...
  [N,Y]=DSTEP(S,NS,NPTS);...
  [I,J]=SIZE(N);... // J and Y are unimportant.  I is used later.
  TIME=N*DT;... // This creates a matrix of time values
END,...
CLEAR J Y N ,...
...//
...// Input strength and perform syntax checks
...//
ZZ=0; WHILE ZZ<1,...
 DISP('INPUT THE STRENGTH OF THE STEP INPUT. STRENGTH MUST HAVE 1'),...
 DISP('COLUMN, AND THE SAME NUMBER OF ROWS AS THE NUMBER OF INPUTS.'),...
 NINPUTS,... // Display the number of inputs for the user
 DISP('EACH ROW OF STRENGTH WILL BE THE STEP SIZE STRENGTH FOR THE'),...
 DISP('RESPECTIVE COMMAND CHANNEL, IE, ROW 1 AFFECTS COMMANDED VALUE'),...
 DISP('1, ETC.'),...
 INQUIRE STRENGTH 'STEP SIZE STRENGTH =',...
 [R,C]=SIZE(STRENGTH); IF R<>NINPUTS,DISP('WRONG NUMBER OF ROWS'),...
     ELSE IF C<>1, DISP('WRONG NUMBER OF COLUMNS'),...
       ELSE ZZ=1;...
END,END,END,...
...//
...// End syntax checking for STRENGTH. STRENGTH has to be turned into
...//  a diagonal matrix so that UCM has the proper dimensions.
...//
U= ONES(I,NINPUTS) * DIAGONAL([STRENGTH]);..
END
```

```
//
// End of PLOTHELP function
//
RETF




//[PLOTFLAG]=PLOTLOOP(TIME,Y,PLOTFLAG)
//
//***************************************************************************
//
// This function allows the user to specify individual channels of an
//  output to be plotted.
//
// This function was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (10 Jan)
//***************************************************************************
//
ERASE // Clear the screen
[I,VARIABLES]=SIZE(Y);
NPLOT=0;
DUMMY=0;
WHILE DUMMY<1,...
 IF VARIABLES>1,...
   DISP('THERE IS MORE THAN ONE VARIABLE TO BE PLOTTED.'),...
   DISP('NUMBER OF VARIABLES ARE'),VARIABLES,...
   DISP('ENTER OUTPUT CHANNEL TO BE PLOTTED (ENTER 0 FOR ALL CHANNELS)'),...
   INQUIRE NPLOT 'OUTPUT NUMBER:  ',...
   IF NPLOT>VARIABLES, NPLOT=VARIABLES;END,...
 END,...
...//
...// Let user give a title
...//
 DISP('ENTER THE TITLE, WITH QUOTATION MARKS AT BEGINNING AND END'),...
 INQUIRE TITLE 'ENTER TITLE: ',...
...//
...// Plot the variables
...//
 IF NPLOT=0,...
  PLOT(TIME,Y,['TITLE\',TITLE]),...
 ELSE ...
  PLOT(TIME,Y(:,NPLOT),['TITLE\',TITLE]),...
 END,...
 INQUIRE PL 'IF YOU WANT A HARDCOPY, ENTER 0. OTHERWISE, ENTER 1. ',...
 IF PL = 0, EXECUTE('PLOT.HARDCOPY'),END,...
 ERASE,...
 INQUIRE DUMMY 'DO ANOTHER PLOT FOR THIS OPTION, ENTER 0. OTHERWISE, 1.
',...
END
//
// End of PLOT.LOOP function
//
RETF
```

```
// ANALYZE.YCMD command file
//
//*************************************************************************
//
// The only reason this command file exists is because MATRIXX doesn't
//  like mixing SYSTEM BUILD commands in with regular MATRIXX commands.
//  When I tried putting these three lines of code in the CGTPI.ANALYZE
//  command file, MATRIXX threw-up all over me.
//
// This command file is called by the CGTPI.ANALYZE command file.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (16 Dec)
//*************************************************************************
//
BUILD
Analyze Super-Block
YCMD
//
// End of ANALYZE.YCMD command file
//
RETURN
```

```
// ANALYZE.Ytm command file
//
//*************************************************************************
```

```
//
// The only reason this command file exists is because MATRIXX doesn't
//  like mixing SYSTEM BUILD commands in with regular MATRIXX commands.
//  When I tried putting these three lines of code in the CGTPI.ANALYZE
//  command file, MATRIXX threw-up all over me.
//
// This command file is called by the CGTPI.ANALYZE command file.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (6 Jan)
//***********************************************************************
//
BUILD
Analyze Super-Block
Ytm
//
// End of ANALYZE.Ytm command file
//
RETURN
```

```
// ANALYZE.XTM command file
//
//***********************************************************************
//
// The only reason this command file exists is because MATRIXX doesn't
//  like mixing SYSTEM BUILD commands in with regular MATRIXX commands.
//  When I tried putting these three lines of code in the CGTPI.ANALYZE
//  command file, MATRIXX threw-up all over me.  Here, XTM is the super-
//  block which represents the truth model driven by the control law
//  developed by the CGTPI command file.  These states are NOT the states
//  of the uncontrolled truth model.
//
// This command file is called by the CGTPI.ANALYZE command file.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (30 Nov)
//***********************************************************************
//
BUILD
Analyze Super-Block
XTM
//
// End of ANALYZE.XTM command file
//
RETURN



// ANALYZE.ACT command file
//
//***********************************************************************
//
// The only reason this command file exists is because MATRIXX doesn't
//  like mixing SYSTEM BUILD commands in with regular MATRIXX commands.
//  When I tried putting these three lines of code in the CGTPI.ANALYZE
//  command file, MATRIXX threw-up all over me.
//
// This command file is called by the CGTPI.ANALYZE command file.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (11 Jan)
//*****************************************************************************
//
BUILD
Analyze Super-Block
ACT
//
// End of ANALYZE.ACT command file
//
RETURN
```

```
// CONTROL command file
//
//***************************************************************************
//
// This command file accesses MATRIXX SYSTEM BUILD and puts together
//  a series of super-blocks which generate the incremental form
//  CGT/PI control law.  It calls several command files to build the
//  required super-blocks.
//
// Command files called: KZYCM.BLOCKS, KZYCM.CONNECT2, KZYCM.CONNECT1,
//   KZYTM.BLOCKS, KZYTM.CONNECT2, KZYTM.CONNECT1,
//   GAIN1.BLOCKS, GAIN1.INTCON2, GAIN1.INTCON1, GAIN1.EXTCON2, GAIN1.EXTCON1
//   GAIN3.BLOCKS, GAIN3.INTCON2, GAIN3.INTCON1, GAIN3.EXTCON2, GAIN3.EXTCON1
//   LAW.BLOCKS, LAW.SUM, LAW.CONNECT2, LAW.CONNECT1
//   YTM.2, YTM.1, XTMD.2, XTMD.1, YCMD.1, YCMD.2
//
// Because of the possible addition of actuator dynamics, the truth model
//  is broken up for the simulation into two super-blocks, one representing
//  the actuators, and one representing the rest of the system.  The
//  actuator dynamics, as well as any position or rate limits imposed
on the
//  actuators, are included in super-block ACT.
//
//
// Written by Captain Steve Payson, 1988.
// Version 1.0  (2 Feb)
//***************************************************************************
//
```

```
// Prompt the user for the design and command model initial conditions
//
J=0;
WHILE J<>NSTM, DISP('INPUT THE TRUTH MODEL STATE VECTOR INITIAL'),...
  DISP('CONDITIONS. YOU MUST HAVE 1 COLUMN AND THE SAME NUMBER OF ROWS'),...
  DISP('AS THE NUMBER OF STATES'),NSTM,...
  INQUIRE XTMO 'XTMO = ',...
  [J,I]=SIZE(XTMO);...
  IF I<>1, J=0,END,... // J is the flag, so if the user didn't input 1
...                      //  column, J is set to zero and this routine repeats
END
//
J=0;
WHILE J<>NSCM, DISP('INPUT THE COMMAND MODEL STATE VECTOR INITIAL'),...
  DISP('CONDITIONS. YOU MUST HAVE 1 COLUMN AND THE SAME NUMBER OF ROWS'),...
  DISP('AS THE NUMBER OF STATES'),NSCM,...
  INQUIRE XCMO 'XCMO = ',...
  [J,I]=SIZE(XCMO);...
  IF I<>1, J=0,END,... // J is the flag, so if the user didn't input 1
...                     //  column, J is set to zero and this routine repeats
END
//
// Define some terms to be used in the ACT super-block, which is built
//  regardless of whether actuator dynamics are present or not.
//  SACTHOT is a State space matrix representing the ACTuator Higher
//  Order Terms, and is square with dimensions of NSACT.
//  SACT1 is a state space matrix representing the
//  actuator first order terms, and is square with dimensions of 2*NINPUTS.
```

```
//
WINDUPFLAG=0;
IF NSACT=0, ... // That is, no actuator dynamics
  SACT1=EYE(NINPUTS);SACTHOT=SACT1;...
ELSE ... // That is, derivatives of actuators exist in the truth model
  SACT1= [ATM((NSTM-NSACT+1):(NSTM-NSACT+NINPUTS),...
              (NSTM-NSACT+1):(NSTM-NSACT+NINPUTS)),...
          BTM((NSTM-NSACT+1):(NSTM-NSACT+NINPUTS),:);...
          EYE(NINPUTS),0*EYE(NINPUTS)];...
  IF NSACT=NINPUTS,... // That is, only one derivative of actuators
    SACTHOT=EYE(NINPUTS);...
  ELSE
    SACTHOT= [ATM((NSTM-NSACT+NINPUTS+1):NSTM,(NSTM-NSACT+NINPUTS+1):NSTM),..
              BTM((NSTM-NSACT+NINPUTS+1):NSTM,:);...
              EYE(NINPUTS),0*ONES(NINPUTS,NSACT-NINPUTS)];...
  END,...
END
//
// Allow the user to define position and rate limits of actuators, and
//  turn antiwindup compensation on.
//
DISP('IF YOU WISH TO IMPOSE ACTUATOR LIMITS,')
INQUIRE LIMFLAG 'ENTER A 0. OTHERWISE, ENTER 1 '
IF LIMFLAG=1, DISP('ANY FORMER VALUES OF POSITION AND RATE LIMITS ARE'),...
 DISP('NOW BEING ERASED.'),...
```

```
...//
...// If no limits desired, set them to an arbitrary value for use
...//  in ACT.BLOCKS*. They will be removed in ACT.NOLIMITS.
...//
 PLOWLIM=ONES(1,NINPUTS);PUPLIM=2*PLOWLIM;RLOWLIM=PLOWLIM;RUPLIM=PUPLIM;...
...//
ELSE,...// that is, LIMFLAG=0
 DUMMY=0;...
 WHILE DUMMY<1,...
  DISP('YOU MAY NOW ENTER ACTUATOR POSITION AND RATE LIMITS.'),...
  DISP('FIRST, ENTER THE POSITION LOWER LIMIT AS A ROW VECTOR, WITH THE'),...
  DISP('NUMBER OF COLUMNS EQUAL TO THE NUMBER OF INPUTS,'),NINPUTS,...
  INQUIRE PLOWLIM 'POSITION LOWER LIMITS (PLOWLIM)= ',...
  [I,J]=SIZE(PLOWLIM); IF I>1, DISP('WRONG NUMBER OF ROWS. TRY AGAIN.'),...
   ELSE IF J=NINPUTS, DUMMY=1;...
    ELSE DISP('WRONG NUMBER OF COLUMNS. TRY AGAIN.'),...
  END,END,...
 END,...
 DUMMY=0;...
 WHILE DUMMY<1,...
  DISP('NOW ENTER THE POSITION UPPER LIMIT AS A ROW VECTOR, WITH THE'),...
  DISP('NUMBER OF COLUMNS EQUAL TO THE NUMBER OF INPUTS,'),NINPUTS,...
  INQUIRE PUPLIM 'POSITION UPPER LIMITS (PUPLIM)= ',...
  [I,J]=SIZE(PUPLIM); IF I>1, DISP('WRONG NUMBER OF ROWS. TRY AGAIN.'),...
   ELSE IF J=NINPUTS, DUMMY=1;...
    ELSE DISP('WRONG NUMBER OF COLUMNS. TRY AGAIN.'),...
  END,END,...
 END,... // end of generating position limits
```

```
...//
...// Generate rate limits.  If no actuator dynamics exist, the limits
will
...//  the same as the position limits, which is valid because they are
...//  cascaded together.
...//
 IF NSACT=0, RLOWLIM=PLOWLIM; RUPLIM=PUPLIM;...
 ELSE ...
  DUMMY=0;...
  WHILE DUMMY<1,...
   DISP('NOW ENTER THE RATE LOWER LIMIT AS A ROW VECTOR, WITH THE'),...
   DISP('NUMBER OF COLUMNS EQUAL TO THE NUMBER OF INPUTS,'),NINPUTS,...
   INQUIRE RLOWLIM 'RATE LOWER LIMITS (RLOWLIM)= ',...
   [I,J]=SIZE(RLOWLIM); IF I>1, DISP('WRONG NUMBER OF ROWS. TRY AGAIN.'),...
    ELSE IF J=NINPUTS, DUMMY=1;...
     ELSE DISP('WRONG NUMBER OF COLUMNS. TRY AGAIN.'),...
   END,END,...
  END,...
  DUMMY=0;...
  WHILE DUMMY<1,...
   DISP('NOW ENTER THE RATE UPPER LIMIT AS A ROW VECTOR, WITH THE'),...
   DISP('NUMBER OF COLUMNS EQUAL TO THE NUMBER OF INPUTS,'),NINPUTS,...
   INQUIRE RUPLIM 'RATE UPPER LIMITS (RUPLIM)= ',...
   [I,J]=SIZE(RUPLIM); IF I>1, DISP('WRONG NUMBER OF ROWS. TRY AGAIN.'),...
    ELSE IF J=NINPUTS, DUMMY=1;...
     ELSE DISP('WRONG NUMBER OF COLUMNS. TRY AGAIN.'),...
   END,END,...
  END,...
 END,... // End of generating rate limits
 DISP('IF YOU WANT TO TURN ANTIWINDUP COMPENSATION ON,'),...
 INQUIRE WINDUPFLAG 'ENTER 1. OTHERWISE, ENTER 0 : ',...
END
//
// End of routine allowing actuator limits and antiwindup compensation
//
CLEAR I J
//
// TMNMEAS, TMNINPUTS, and TMNOUTPUTS are used in the following command
//  files, but at the present time they are all equal to NINPUTS.
//  Define these variables now, but clear them at the end of this
//  command file to reduce the number of names.
//
I=NINPUTS;TMNOUTPUTS=I;TMNINPUTS=I;TMNMEAS=NMEAS;
CLEAR I
//
DISP('THIS WILL TAKE A WHILE. PLEASE BE PATIENT.')
//
```

```
//**********************************************************************
// Generate the control law.  This involves building several blocks.
//  First, discretize the truth model.
//
STM=[ATM,BTM;CTM,DTM];
[STMD]=DISCRETIZE(STM,NSTM,DT);
[ATMD,BTMD,CTM,DTM]=SPLIT(STMD,NSTM);
// Generate the internal blocks of super-blocks KZYCM and KZYTM
//
EXECUTE('KZYCM.BLOCKS')
EXECUTE('KZYTM.BLOCKS')
//
// Generate the internal and external connections of super-blocks
// KZYCM and KZYTM
//
IF NINPUTS>1, EXECUTE('KZYCM.CONNECT2'), EXECUTE('KZYTM.CONNECT2'),...
 ELSE EXECUTE('KZYCM.CONNECT1'), EXECUTE('KZYTM.CONNECT1'),END
//
// Generate super-block GAIN1
//
EXECUTE('GAIN1.BLOCKS')
FOR N=1:NSTM, EXECUTE('GAIN1.SUM'), END
IF NSTM>1, EXECUTE('GAIN1.INTCON2'),...
  ELSE EXECUTE('GAIN1.INTCON1'),END
IF NINPUTS>1, EXECUTE('GAIN1.EXTCON2'),...
  ELSE EXECUTE('GAIN1.EXTCON1'), END
//
// Generate super-block GAIN3
```

```
//
EXECUTE('GAIN3.BLOCKS')
FOR N=1:NSCM, EXECUTE('GAIN3.SUM'), END
IF NSCM>1, EXECUTE('GAIN3.INTCON2'),...
  ELSE EXECUTE('GAIN3.INTCON1'),END
IF NINPUTS>1, EXECUTE('GAIN3.EXTCON2'),...
  ELSE EXECUTE('GAIN3.EXTCON1'), END
//
// Generate super-block BLOCK1, which is the combination of GAIN3 an
//  KZYCM.
//
EXECUTE('BLOCK1.BLOCKS')
IF NINPUTS>1, EXECUTE('BLOCK1.CONNECT2'),...
 ELSE EXECUTE('BLOCK1.CONNECT1'), END
FOR N=1:NINPUTS, EXECUTE('BLOCK1.SUM'), END
//
// Generate super-blocks LAW, YTM, and ACT.  LAW is the discrete-time
control
//  law, while ACT represents actuator dynamics and limits.
//  YTM is the combination of the control law and the continuous
//  truth model (including ACT), that is, the controlled system's outputs.
//
DUMMY=NSTM-NSACT; // used in YTM.1 and YTM.2 command files
IF NSACT=0, EXECUTE('ACT.BLOCKS0')
IF NSACT=NINPUTS, EXECUTE('ACT.BLOCKS1')
IF NSACT>NINPUTS, EXECUTE('ACT.BLOCKS2')
IF LIMFLAG=1, EXECUTE('ACT.NOLIMITS') // turn off limits
IF WINDUPFLAG=1, EXECUTE('LAW.BLOCKS1'),...
```

```
 ELSE  EXECUTE('LAW.BLOCKS0'), END
FOR N=1:NINPUTS, EXECUTE('LAW.SUM'), END
//
// Define the state space matrices used in YTM.* and XTM.*
//
// Case of no actuator dynamics
IF NSACT=0,...
  SY=[ATM,BTM;CTM,DTM];SX=[ATM,BTM;EYE(NSTM),0*ONES(NSTM,NINPUTS)];
// Case of only first order actuator dynamics
IF NSACT=NINPUTS, SY=[ATM(1:DUMMY,:);CTM(1:NINPUTS,1:DUMMY),DTM];...
      SX=[ATM(1:DUMMY,:);EYE(DUMMY),0*ONES(DUMMY,NINPUTS)];
// Case of higher than first order actuator dynamics
IF NSACT>NINPUTS, SY=[ATM(1:DUMMY,1:DUMMY+NINPUTS);...
                      CTM(1:NINPUTS,1:DUMMY),DTM];...
      SX=[ATM(1:DUMMY,1:DUMMY+NINPUTS);...
         EYE(DUMMY),0*ONES(DUMMY,NINPUTS)];
//
// Finish building LAW and ACT, and build super-block YTM.
//
IF NINPUTS>1, EXECUTE('LAW.CONNECT2'),EXECUTE('YTM.2'),...
   EXECUTE('ACT.CONNECT2'),...
 ELSE EXECUTE('LAW.CONNECT1'),EXECUTE('YTM.1'),...
   EXECUTE('ACT.CONNECT1'),...
END
//
// Generate super-block XTM, the controlled system's state response.
//  Also generate super-block YCMD, the command model's outputs (for use
//  by CGTPI.ANALYZE 'PLOT COM MOD OUTPUT' option).
```

```
//
IF NINPUTS>1, EXECUTE('XTM.2'), EXECUTE('YCMD.2'),...
  ELSE EXECUTE('XTM.1'), EXECUTE('YCMD.1'),END
//
// Turn SYSTEM BUILD's PLOT back on, because it was turned off in KZYCM.BLOCK
//
BUILD,Commands,Set Auto PLOT ON, Top,Mat
CLEAR LIMFLAG WINDUPFLAG SX SY SACTHOT SACT1 DUMMY TMNOUTPUTS TMNMEAS
CLEAR TMNINPUTS
//
// End of CONTROL command file
//
RETURN
```

```
// KZYCM.BLOCKS command file
//************************************************************************
//
// This command file is part one of three command files which generate
the
//  output from the command model, Ycm, time delays it (Ycm(t_i-1)), and
//  multiplies it by the gain KZ. It is written using MATRIXX SYSTEM BUILD
//  commands.  The reason the command file is broken into three parts
is
//  that SYSTEM BUILD querries the user for an input if connections between
//  blocks involve vectors with dimension greater than 1.  This part of
//  the program generates the blocks inside the super-block KZYCM, while
//  two other command files called KZYCM.CONNECT1 or KZYCM.CONNECT2
//  connect the blocks, depending on the dimension of the connections.
//
// The name of this command file represents the fact that the output of
//  it is the gain  KZ multiplied by Ycm, and that only the BLOCKS of
the
//  super-model are generated.  The input is Ucm and the output is
//  KZ*Ycm(t_i-1).
//
// This command file is called by the CONTROL command file.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (18 Nov)
//************************************************************************
//
 BUILD
 Command // change menu
 Set Auto PLOT OFF // turn graphics off
 Top // change menu
 Edit Super-Block
 KZYCM // name of super-block
 DT // sample period
 0 // first sample time
//
 Define Block
 1 // block loation
 Dynamic Systems
 State-Space System
 Block Name
 COMMOD // this block is the state space matrix [a,b;c,d]
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 NSCM
 Parameter Entry
 [ACMD,BCMD;CCM,DCM]
 N // initial states not zero
 XCMO // command model initial conditions
//
 Define Block
 2 // block location
 Dynamic Systems
 Time Delay: exp(-kTs)
 Block Name
 DELAY
 Number of Outputs
 NINPUTS
 Parameter Entry
 DT // amount of time delay
//
 Define Block
 3 // block location
 Dynamic Systems
 State-Space System
 Block Name
 KZ
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 0
 Parameter Entry
 KZ
//
 TOP
 MAT
//
// End of KZYCM.BLOCKS command file
//
```

```
RETURN



// KZYCM.CONNECT2 command file
//*****************************************************************************
//
// This command file is part 3 of three command files which generate the
//  output from the command model, Ycm, time delays it (Ycm(t_i-1)), and
//  multiplies it by the gain KZ.  It is written using MATRIXX SYSTEM
BUILD
//  commands.  The reason the command file is broken into three parts
is
//  that SYSTEM BUILD querries the user for an input if connections between
//  blocks involve vectors with dimension greater than 1.  This part of
the
//  program connects the blocks if the dimension of the connections is
//  greater than 1.  KZYCM.BLOCKS built the blocks being connected, while
//  KZYCM.CONNECT1 connects the blocks if the dimension is less than 1.
//  NINPUTS (the number of controls and outputs) is the flag for the dimensio
//  of the connections.
//
// The name of this command file represents the fact that the output of
it
//  is the gain KZ multiplied by Ycm, and that the blocks are CONNECTed
//  here if there dimension is greater than 1.  The input is Ucm and the
//  output is KZ*Ycm(t_i-1).
//
```

```
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (14 Nov)
//************************************************************************
//
 BUILD
 Edit Super-Block
 KZYCM // name of super-block
 0 // same sample period as KZYDM.BLOCKS used when creating this super-block.
// first sample time is not required when editing an existing super-block.
//
//
// Now connect the blocks; block 1 to 2, and 2 to 3
//
 Connect Blocks
 Internal Path
 1
 2
 Y // connection is simple
 Internal Path
 2
 3
 Y // connection is simple
 External Input
 NINPUTS // number of inputs to the super-block
 1 // external inputs come into block 1
 Y // connection is simple
 External Output
 NINPUTS // number of outputs from the super-block
 3 // external outputs come from block 3
 Y // connection is simple
 TOP
 MAT
//
// End of KZYCM.CONNECT2 command file
//
RETURN




// KZYCM.CONNECT1 command file
//****************************************************************************
//
// This command file is part 2 of three command files which generate the
//  output from the command model, Ycm, time delays it (Ycm(t_i-1)), and
//  multiplies it by the gain KZ.  It is written using MATRIXX SYSTEM
BUILD
//  commands.  The reason the command file is broken into three parts
is
//  that SYSTEM BUILD querries the user for an input if connections between
//  blocks involve vectors with dimension greater than 1.  This part of
the
//  program connects the blocks if the dimension of the connections is
1.
//  KZYCM.BLOCKS built the blocks being connected, while KZYCM.CONNECT2
//  connects the blocks if the dimension is greater than 1.  NINPUTS (
the
//  number of controls and outputs) is the flag for the dimension of the
//  connections.
//
// The name of this command file represents the fact that the output of
it
//  is the gain KZ multiplied by Ycm, and that the blocks are CONNECTed
//  here if their dimension is 1.  The input is Udm and the output is
//  KZ*Ycm(t_i-1).
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (14 Nov)
//*************************************************************************
//
 BUILD
 Edit Super-Block
 KZYCM // name of super-block
 0 // same sample period as KZYDM.BLOCKS used when creating this super-block.
// first sample time is not required when editing an existing super-block.
//
//
// Now connect the blocks; block 1 to 2, and 2 to 3
//
 Connect Blocks
 Internal Path
 1
 2
 Internal Path
 2
 3
 External Input
 NINPUTS // number of inputs to the super-block
 1 // external inputs come into block 1
 External Output
 NINPUTS // number of outputs from the super-block
 3 // external outputs come from block 3
 TOP
 MAT
//
// End of KZYCM.CONNECT1 command file
//
RETURN
```

```
// KZYTM.BLOCKS command file
//*************************************************************************
//
// This command file is part one of three command files which generate
the
//  output from the truth model, Ytm, time delays it (Ytm(t_i-1)), and
//  multiplies it by the gain KZ. It is written using MATRIXX SYSTEM BUILD
//  commands.  The reason the command file is broken into three parts
is
//  that SYSTEM BUILD querries the user for an input if connections between
//  blocks involve vectors with dimension greater than 1.  This part of
//  the program generates the blocks inside the super-block KZYTM, while
//  two other command files called KZYTM.CONNECT1 or KZYTM.CONNECT2
//  connect the blocks, depending on the dimension of the connections.
//
// The name of this command file represents the fact that the output of
//  it is the gain  KZ multiplied by Ytm, and that only the BLOCKS of
the
//  super-model are generated.  The input is Udm and the output is
//  KZ*Ytm(t_i-1).
//
// This command file is called by the CONTROL command file
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (13 Jan)
//*************************************************************************
//
 BUILD
```

```
Edit Super-Block
KZYTM // name of super-block
DT // sample period
0 // first sample time
//
Define Block
1 // block location
Dynamic Systems
State-Space System
Block Name
TRUMOD // this block is the state space matrix [a,b;c,d]
Number of Inputs
NINPUTS
Number of Outputs
NINPUTS
Number of States
NSTM
Parameter Entry
[ATMD,BTMD;CTM,DTM]
N // initial states not zero
XTMO // design model initial conditions
//
Define Block
2 // block location
Dynamic Systems
Time Delay: exp(-kTs)
Block Name
DELAY
```

```
 Number of Outputs
 NINPUTS
 Parameter Entry
 DT // amount of time delay
//
 Define Block
 3 // block location
 Dynamic Systems
 State-Space System
 Block Name
 KZ
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 0
 Parameter Entry
 KZ
//
 TOP
 MAT
//
// End of KZYTM.BLOCKS command file
//
RETURN
```

```
// KZYTM.CONNECT2 command file
//*************************************************************************
//
// This command file is part 3 of three command files which generate the
//  output from the truth model, Ytm, time delays it (Ytm(t_i-1)), and
//  multiplies it by the gain KZ.  It is written using MATRIXX SYSTEM
BUILD
//  commands.  The reason the command file is broken into three parts
is
//  that SYSTEM BUILD querries the user for an input if connections between
//  blocks involve vectors with dimension greater than 1.  This part of
the
//  program connects the blocks if the dimension of the connections is
//  greater than 1.  KZYTM.BLOCKS built the blocks being connected, while
//  KZYTM.CONNECT1 connects the blocks if the dimension is less than 1.
//  NINPUTS (the number of controls and outputs) is the flag for the dimensio
//  of the connections.
//
// The name of this command file represents the fact that the output of
it
//  is the gain KZ multiplied by Ytm, and that the blocks are CONNECTed
//  here if there dimension is greater than 1.  The input is Udm and the
//  output is KZ*Ytm(t_i-1).
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (13 Jan)
```

```
//***************************************************************************
//
 BUILD
 Edit Super-Block
 KZYTM // name of super-block
 0 // same sample period as KZYTM.BLOCKS used when creating this super-block.
// first sample time is not required when editing an existing super-block.
//
//
// Now connect the blocks; block 1 to 2, and 2 to 3
//
 Connect Blocks
 Internal Path
 1
 2
 Y // connection is simple
 Internal Path
 2
 3
 Y // connection is simple
 External Input
 NINPUTS // number of inputs to the super-block
 1 // external inputs come into block 1
 Y // connection is simple
 External Output
 NINPUTS // number of outputs from the super-block
 3 // external outputs come from block 3
 Y // connection is simple
 TOP
 MAT
//
// End of KZYTM.CONNECT2 command file
//
RETURN




// KZYTM.CONNECT1 command file
//*************************************************************************
//
// This command file is part 2 of three command files which generate the
//  output from the truth model, Ytm, time delays it (Ytm(t_i-1)), and
//  multiplies it by the gain KZ.  It is written using MATRIXX SYSTEM
BUILD
//  commands.  The reason the command file is broken into three parts
is
//  that SYSTEM BUILD querries the user for an input if connections between
//  blocks involve vectors with dimension greater than 1.  This part of
the
//  program connects the blocks if the dimension of the connections is
1.
//  KZYTM.BLOCKS built the blocks being connected, while KZYTM.CONNECT2
//  connects the blocks if the dimension is greater than 1.  NINPUTS (
the
//  number of controls and outputs) is the flag for the dimension of the
//  connections.
//
// The name of this command file represents the fact that the output of
it
//  is the gain KZ multiplied by Ytm, and that the blocks are CONNECTed
//  here if their dimension is 1.  The input is Udm and the output is
//  KZ*Ytm(t_i-1).
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (13 Jan)
//*************************************************************************
//
 BUILD
 Edit Super-Block
 KZYTM // name of super-block
 0 // same sample period as KZYTM.BLOCKS used when creating this super-block.
// first sample time is not required when editing an existing super-block.
//
//
// Now connect the blocks; block 1 to 2, and 2 to 3
//
 Connect Blocks
 Internal Path
 1
 2
 Internal Path
 2
 3
 External Input
 NINPUTS // number of inputs to the super-block
 1 // external inputs come into block 1
 External Output
 NINPUTS // number of outputs from the super-block
 3 // external outputs come from block 3
 TOP
 MAT
//
// End of KZYTM.CONNECT1 command file
//
RETURN
```

```
// GAIN1.BLOCKS command file
//
//*****************************************************************************
//
// This command file  is part 1 of 6 which generates the first gain of
//  equation 14-267 in Maybeck's Volume III, that is, KX*[Xdm(t_i)-Xdm(t_i-1)
//  It is written using MATRIXX SYSTEM BUILD commands.  The input to the
super-
//  block is Udm.  The name of
//  this command file is meant to represent the first gain or equation
//  14-267 in Maybeck's Volume III, and that it generates the internal
//  BLOCKS of the super-block.  Other command files connect the blocks.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (13 Jan)
//*****************************************************************************
//
 BUILD
 Edit Super-Block
 GAIN1 // name of superblock
 DT // sample period
 0 // first sample time
//
 Define block
 1 // block location. This block generates Xdm (discrete).
 Dynamic Systems
 State-Space System
 Block Name
 XTMD
 Number of Inputs
 NINPUTS
 Number of outputs
 NSTM
 Number of States
 NSTM
 Parameter Entry
 [ATMD,BTMD;EYE(NSTM),0*ONES(NSTM,NINPUTS)]
 N // initial states not zero
 XTMO // design model initial conditions
//
 Define Block
 2 // block location.  This block will generate Xdm(t_i-1)
 Dynamic Systems
 Time Delay: exp(-kTs)
 Block Name
 DELAY
 Number of Outputs
 NSTM
 Parameter Entry
 DT // amount of time delay
//
 Define Block
 3 // block location
 Algebraic Equations
 Sum of Vectors
 Number of Outputs
 NSTM
 Parameter Entry
 2 // number of input vectors
 -1,1 // sign on each input vector
//
 Define Block
 4 // block location
 Dynamic Systems
 State-Space System
 Block Name
 TTM
 Number of Inputs
 NSTM
 Number of Outputs
 NSDM
 Number of States
 0
 Parameter Entry
 TTM // transformation matrix
//
 Define Block
 5 // block location
 Dynamic Systems
 State-Space System
 Block Name
 KX
 Number of Inputs
 NSDM
 Number of Outputs
 NINPUTS
 Number of States
 0
 Parameter Entry
 KX // KX has NSDM columns, NINPUTS rows, which is why the TTM matrix
is needed
//
 Top
 Mat
//
// End of GAIN1.BLOCKS command file
//
RETURN




// GAIN1.SUM command file
//
//***************************************************************************
//
// This command file  is part 2 of 6 which generates the first gain of
//  equation 14-267 in Maybeck's Volume III, that is, KX*[Xtm(t_i)-Xtm(t_i-1)
//  It is written using MATRIXX SYSTEM BUILD commands.  The input to the
super-
//  block is Udm.  The name of
//  this command file is meant to represent the first gain of equation
//  14-267 in Maybeck's Volume III, and that it deals with the SUMming
//  junction.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (13 Jan)
//*******************************************************************************
//
 BUILD
 Edit Super-Block
 GAIN1 // name of superblock
 0 // same sample period as GAIN1.BLOCKS used when generating this super-bloc
// first sample time is not required when editing an existing super-block
//
//
 Connect Blocks // connect block 1 to 2 and 2 to 3
 Internal Path
 2
 3
 N,N // a connection
 0,0 // end connection mode
 Internal Path
 1
 3
 N,(N+NSTM) // a connection
 0,0 // end connection mode
//
// The above connections to the summing block result in x(t_i) - x(t_i-1)
//
```

```
 Top
 Mat
//
// End of GAIN1.SUM command file
//
RETURN




// GAIN1.INTCON2 command file
//
//*****************************************************************************
//
// This command file  is part 3 of 6 which generates the first gain of
//  equation 14-267 in Maybeck's Volume III, that is, KX*[Xtm(t_i)-Xtm(t_i-1)
//  It is written using MATRIXX SYSTEM BUILD commands.  The input to the
super-
//  block is Udm.  The name of
//  this command file is meant to represent the first gain of equation
//  14-267 in Maybeck's Volume III, and that it  deals with INTernal
//  CONnections with dimension 2 or greater. NSCM is the flag for this
//  command file, because that is the dimension of the internal connections.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (13 Jan)
//*****************************************************************************
//
```

```
 BUILD
 Edit Super-Block
 GAIN1 // name of superblock
 0 // same sample period as GAIN1.BLOCKS used when generating this super-bloc
// first sample time is not required when editing an existing super-block
//
//
 Connect Blocks // connect block 1 to 2 and 2 to 3
 Internal Path
 1
 2
 Y // simple connection
 Internal Path
 3
 4
 Y // simple connection
//
 Internal Path
 4
 5
 Y // simple connection
//
 Top
 Mat
//
// End of GAIN1.INTCON2 command file
//
RETURN
```

```
// GAIN1.INTCON1 command file
//
//***************************************************************************
//
// This command file  is part 4 of 6 which generates the first gain of
//  equation 14-267 in Maybeck's Volume III, that is, KX*[Xdm(t_i)-Xdm(t_i-1)
//  It is written using MATRIXX SYSTEM BUILD commands.  The input to the
super-
//  block is UDM.  The name of
//  this command file is meant to represent the first gain of equation
//  14-267 in Maybeck's Volume III, and that it  deals with INTernal
//  CONnections with dimension 1. NSCM is the flag for this command file,
//  because that is the dimension of the internal connections.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (15 Nov)
//***************************************************************************
//
 BUILD
 Edit Super-Block
 GAIN1 // name of superblock
 0 // same sample period as GAIN1.BLOCKS used when generating this super-bloc
// first sample time is not required when editing an existing super-block
//
```

```
//
 Connect Blocks // connect block 1 to 2 and 2 to 3
 Internal Path
 1
 2
//
 Internal Path
 3
 4
//
 Internal Path
 4
 5
//
 Top
 Mat
//
// End of GAIN1.INTCON2 command file
//
RETURN
```

```
// GAIN1.EXTCON2 command file
//
//*****************************************************************************
//
```

```
// This command file  is part 5 of 6 which generates the first gain of
//  equation 14-267 in Maybeck's Volume III, that is, KX*[Xtm(t_i)-Xtm(t_i-1)
//  It is written using MATRIXX SYSTEM BUILD commands.  The input to the
super-
//  block is Udm.  The name of
//  this command file is meant to represent the first gain of equation
//  14-267 in Maybeck's Volume III, and that it deals with EXTernal CONnectio
//  with dimension 2 or greater.  NINPUTS is the flag for this command
file,
//  because that is the dimension of the external connections.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (13 Jan)
//*********************************************************************
//
 BUILD
 Edit Super-Block
 GAIN1 // name of superblock
 0 // same sample period as GAIN1.BLOCKS used when generating this super-bloc
// first sample time is not required when editing an existing super-block
//
//
 Connect Blocks // connect block 1 to 2 and 2 to 3
 External Input
 NINPUTS
 1 // external input goes to block 1, which generates Xdm
 Y // connection is simple
 External Output
 NINPUTS
 5 // external output from block 5
 Y // connection is simple
//
 Top
 Mat
//
// End of GAIN1.EXTCON2 command file
//
RETURN




// GAIN1.EXTCON1 command file
//
//****************************************************************************
//
// This command file is part 6 of 6 which generates the first gain of
//  equation 14-267 in Maybeck's Volume III, that is, KX*[Xtm(t_i)-Xtm(t_i-1)
//  It is written using MATRIXX SYSTEM BUILD commands.  The input to the
super-
//  block is Udm.  The name of
//  this command file is meant to represent the first gain of equation
//  14-267 in Maybeck's Volume III, and that it deals with EXTernal CONnectio
//  with dimension 1.  NINPUTS is the flag for this command file, because
//  that is the dimension of the external connections.
//
```

```
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (15 Nov)
//*****************************************************************************
//
 BUILD
 Edit Super-Block
 GAIN1 // name of superblock
 0 // same sample period as GAIN1.BLOCKS used when generating this super-bloc
// first sample time is not required when editing an existing super-block
//
//
 Connect Blocks // connect block 1 to 2 and 2 to 3
 External Input
 NINPUTS
 1 // external input goes to block 1, which generates Xdm
 External Output
 NINPUTS
 5 // external output from block 5
//
 Top
 Mat
//
// End of GAIN1.EXTCON1 command file
//
RETURN
```

```
// GAIN3.BLOCKS command file
//*************************************************************************
//
// This command file is part 1 of 6 which generates the value
//  [KX*A11+A21][Xcm(t_i)-Xcm(t_i-1)]
//  and is written using MATRIXX SYSTEM BUILD commands.
//  The name of this command file is meant to represent the ouput of
//  this file, which is the third gain of equation 14-267 in Maybeck's
//  Volume III, and that only the internal BLOCKS of the super-block
//  are generated.  Other command files connect the blocks.
//
// This command file was written by Captain Steve Payson
// Version 1.0  (18 Nov)
//*************************************************************************
//
 BUILD
 Edit Super-Block
 GAIN3 // name of super-block
 DT // sample period
 0 // first sample time
//
 Define Block
 1 // block location. This block generates Xcm (discrete).
 Dynamic Systems
 State-Space System
 Block Name
 XCMD
 Number of Inputs
 NINPUTS
 Number of Outputs
 NSCM
 Number of States
 NSCM
 Parameter Entry
 [ACMD,BCMD;EYE(NSCM),0*ONES(NSCM,NINPUTS)]
 N // initial states not zero
 XCMO // command model initial conditions
//
// Define the block which generates Xcm(t_i-1)
//
 Define Block
 2 // block location
 Dynamic Systems
 Time Delay: exp(-kTs)
 Block Name
 DELAY
 Number of Outputs
 NSCM
 Parameter Entry
 DT  // amount of time delay.  End of block 2 definition.
//
// Define the multiplication factor KXM, which is KX*A11+A21
//
 Define Block
 4 // block location
 Dynamic Systems
 State-Space System
 Block Name
 KXM
 Number of Inputs
 NSCM
 Number of Outputs
 NINPUTS
 Number of States
 0
 Parameter Entry
 KXM
//
// Define summing junction
//
 Define Block
 3 // block location
 Algebraic Equations
 Sum of Vectors
 Number of Outputs
 NSCM
 Parameter Entry
 2 // two input vectors
 1,-1 // sign on each input vector
//
 Top
 Mat
//
// End of GAIN3.BLOCKS command file
```

```
//
RETURN




// GAIN3.SUM command file
//*********************************************************************
//
// This command file is part 2 of 6 which generates the value
//  [KX*A11+A21][Xcm(t_i)-Xcm(t_i-1)]
//  and is written using MATRIXX SYSTEM BUILD commands.
//  The name of this command file is meant to represent the ouput of
//  this file, which is the third gain of equation 14-267 in Maybeck's
//  Volume III, and that only the SUMmming junction is being dealt
//  with.  This file is required in order to deal with the dimensions
//  of the vectors being added together, which are of dim NSCM.
//
// This command file was written by Captain Steve Payson
// Version 1.0  (15 Nov)
//*********************************************************************
//
 BUILD
 Edit Super-Block
 GAIN3 // name of super-block
 0 // same sample period as GAIN3.BLOCKS used when creating this super-block
// first sample time is not required when editing an existing super-block
//
```

```
 Connect Blocks
 Internal Path
 1 // the first vector is the output from block 1, and it is added (+1)
 3 // block one connected to block three
 N,N // a connection
 0,0 // end connection mode
 Internal Path
 2 // the second vector is the output from block 2, and it is subtracted
(-1)
 3 // block 2 connected to block 3
 N,(N+NSCM) // a connection
 0,0 // end connection mode
//
// The above connections simply subtract vector 2 from vector 1, which
//  results in Xcm(t_i)-Xcm(t_i-1)
//
 Top
 Mat
//
// End of GAIN3.SUM command file
//
RETURN



// GAIN3.INTCON2 command file
//***********************************************************************
```

```
//
// This command file is part 3 of 6 which generates the value
//  [KX*A11+A21][Xcm(t_i)-Xcm(t_i-1)]
//  and is written using MATRIXX SYSTEM BUILD commands.
//  The name of this command file is meant to represent the ouput of
//  this file, which is the third gain of equation 14-267 in Maybeck's
//  Volume III, and that it deals with INTernal CONnections with
//  dimension 2 or more.  NSCM is the flag for this command file, because
//  that is the dimension of the internal connections.
//
// This command file was written by Captain Steve Payson
// Version 1.0  (15 Nov)
//***********************************************************************
//
 BUILD
 Edit Super-Block
 GAIN3 // name of super-block
 0 // same sample period as GAIN3.BLOCKS used when creating this super-block
// first sample time is not required when editing an existing super-block
//
 Connect Blocks
 Internal Path
 3 // from block 3, the summing junction
 4 // to block 4, which is the gain KX*A11+A21
 Y // the connection is simple
 Internal Path  // connect block 1 to block 2
 1
 2
 Y // the connection is simple
//
 Top
 Mat
//
// End of GAIN3.INTCON2 command file
//
RETURN




// GAIN3.INTCON1 command file
//************************************************************************
//
// This command file is part 4 of 6 which generates the value
//  [KX*A11+A21][Xcm(t_i)-Xcm(t_i-1)]
//  and is written using MATRIXX SYSTEM BUILD commands.
//  The name of this command file is meant to represent the ouput of
//  this file, which is the third gain of equation 14-267 in Maybeck's
//  Volume III, and that it deals with INTernal CONnections with
//  dimension 1.  NSCM is the flag for this command file, because
//  that is the dimension of the internal connections.
//
// This command file was written by Captain Steve Payson
// Version 1.0  (15 Nov)
//************************************************************************
//
```

```
 BUILD
 Edit Super-Block
 GAIN3 // name of super-block
 0 // same sample period as GAIN3.BLOCKS used when creating this super-block
// first sample time is not required when editing an existing super-block
//
 Connect Blocks
 Internal Path
 3 // from block 3, the summing junction
 4 // to block 4, which is the gain KX*A11+A21
 Internal Path  // connect block 1 to block 2
 1
 2
//
 Top
 Mat
//
// End of GAIN3.INTCON1 command file
//
RETURN



// GAIN3.EXTCON2 command file
//*********************************************************************
//
// This command file is part 5 of 6 which generates the value
//  [KX*A11+A21][Xcm(t_i)-Xcm(t_i-1)]
//  and is written using MATRIXX SYSTEM BUILD commands.
//  The name of this command file is meant to represent the ouput of
//  this file, which is the third gain of equation 14-267 in Maybeck's
//  Volume III, and that it deals with EXTernal CONnections with
//  dimension 2 or greater. NINPUTS is the flag for this command file,
//  because that is the dimension of the internal connections.
//
// This command file was written by Captain Steve Payson
// Version 1.0  (15 Nov)
//*********************************************************************
//
 BUILD
 Edit Super-Block
 GAIN3 // name of super-block
 0 // same sample period as GAIN3.BLOCKS used when creating this super-block
// first sample time is not required when editing an existing super-block
//
 Connect Blocks
 External Input
 NINPUTS // the number of inputs to the super-block
 1 // external inputs go to block 1
 Y // connection is simple
 External Output
 NINPUTS // the number of outputs from the super-block
 4 // external outputs are from block 4
 Y // connection is simple
//
```

```
 Top
 Mat
//
// End of GAIN3.EXTCON2 command file
//
RETURN




// GAIN3.EXTCON1 command file
//**********************************************************************
//
// This command file is part 6 of 6 which generates the value
//  [KX*A11+A21][Xcm(t_i)-Xcm(t_i-1)]
//  and is written using MATRIXX SYSTEM BUILD commands.
//  The name of this command file is meant to represent the ouput of
//  this file, which is the third gain of equation 14-267 in Maybeck's
//  Volume III, and that it deals with EXTernal CONnections with
//  dimension 1. NINPUTS is the flag for this command file, because
//  that is the dimension of the internal connections.
//
// This command file was written by Captain Steve Payson
// Version 1.0  (15 Nov)
//**********************************************************************
//
 BUILD
 Edit Super-Block
 GAIN3 // name of super-block
 0 // same sample period as GAIN3.BLOCKS used when creating this super-block
// first sample time is not required when editing an existing super-block
//
 Connect Blocks
 External Input
 NINPUTS // the number of inputs to the super-block
 1 // external inputs go to block 1
 External Output
 NINPUTS // the number of outputs from the super-block
 4 // external outputs are from block 4
//
 Top
 Mat
//
// End of GAIN3.EXTCON1 command file
//
RETURN
```

```
// BLOCK1.BLOCKS command file
//
//*********************************************************************
//
// This command file is part 1 of 4 which generates super-block BLOCK1
for
//  the closed loop CGT/PI/KF simulation.  It uses MATRIXX SYSTEM BUILD
//  commands.  BLOCK1 is part of super-block LAWF.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (28 Jan)
//*********************************************************************
//
 BUILD
 Command // change menu
 Set Auto PLOT OFF // turn graphics off
 Top // change menu
 Edit Super-Block
 BLOCK1 // name of super block
 DT // sample period
 0  // first sample time
//
 Define Block
 1 // block location
 Super-Block
 Block Name
 GAIN3
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 2
 Super-Block
 Block Name
 KZYCM
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 3
 Algebraic Equations
 Sum of Vectors
 Number of Outputs
 NINPUTS
 Parameter Entry
 2
 1, 1
//
 TOP
 MAT
```

```
//
// End of BLOCK1.BLOCKS command file
//
RETURN




// BLOCK1.CONNECT2 command file
//
//***************************************************************************
//
// This command file is part 3 of 4 which generates super-block BLOCK1
for
//  the closed loop CGT/PI/KF simulation.  It uses MATRIXX SYSTEM BUILD
//  commands.  BLOCK1 is part of super-block LAWF.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (28 Jan)
//***************************************************************************
//
 BUILD
 Edit Super-Block
 BLOCK1 // name of super-block
 0 // sample period  same as in BLOCK1.BLOCKS
//
 Connect Blocks
 External Input
 NINPUTS // dimension of external input vector
 1  // external input comes into block 1
 Y  // the connection is simple
 External Input
 NINPUTS
 2  // external input also comes into block 2
 Y
 External Output
 NINPUTS // dimension of external output vector
 3 // external output comes from block 3, the summing junction
 Y
//
 TOP
 MAT
//
// End of BLOCK1.CONNECT2 command file
//
RETURN




// BLOCK1.CONNECT1 command file
//
//****************************************************************************
//
// This command file is part 4 of 4 which generates super-block BLOCK1
for
```

```
//  the closed loop CGT/PI/KF simulation.  It uses MATRIXX SYSTEM BUILD
//  commands.  BLOCK1 is part of super-block LAWF.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (28 Jan)
//***************************************************************************
//
 BUILD
 Edit Super-Block
 BLOCK1 // name of super-block
 DT // sample period same as in BLOCK1.BLOCKS
//
 Connect Blocks
 External Input
 NINPUTS // dimension of external input vector
 1  // external input comes into block 1
 External Input
 NINPUTS
 2  // external input also comes into block 2
 External Output
 NINPUTS // dimension of external output vector
 3 // external output comes from block 3, the summing junction
//
 TOP
 MAT
//
// End of BLOCK1.CONNECT1 command file
//
```

```
RETURN




// BLOCK1.SUM command file
//
//*****************************************************************************
//
// This command file is part 2 of 4 which generates super-block BLOCK1
for
//  the closed loop CGT/PI/KF simulation.  It uses MATRIXX SYSTEM BUILD
//  commands.  BLOCK1 is part of super-block LAWF.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (28 Jan)
//*****************************************************************************
//
 BUILD
 Edit Super-Block
 BLOCK1 // name of super-block
 0 // sample period same as in BLOCK1.BLOCKS
//
 Connect Blocks
 Internal Path
 1
 3
 N,N  // a connection
 0,0  // end connection mode
 Internal Path
 2
 3
 N,(N+NINPUTS)  // a connection
 0,0
//
 TOP
 MAT
//
// End of BLOCK1.SUM command file
//
RETURN
```

```
// LAW.BLOCKSO command file
//
//*************************************************************************
//
// This command file is part 1 of 6 which uses MATRIXX SYSTEM BUILD
//  commands to put together the incremental form CGT/PI control law
//   u(t_i) = u(t_i-1) - KX*[Xtm(t_i)-Xtm(t_i-1)] + KZ*Ycm(t_i-1) -
//             KZ*Ytm(t_i-1) + [KX*A11+A21]*[Xcm(t_i)-Xcm(t_i-1)]
//  which can also be seen in Maybeck's Volume III, equation 14-267.
//  The name of this command file represents the ouput of the super-
//  block, which is the control LAW, and that it generates the internal
//  BLOCKS of the super-block if the WINDUPFLAG=0 (antiwindup compensation
//  is turned off).
//
// The LAW super-block (which is generated by the LAW.* command files)
//  has an input of Ucm (the command model input) and an output of Udm
//  (the design model control law).  It uses the super-blocks GAIN1,
//  GAIN3, KZYCM, and KZYTM.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (1 Feb)
//*************************************************************************
//
 BUILD
 Edit Super-Block
 LAW // name of super-block
 DT // sample period
 0 // first sample time
```

```
//
 Define Block
 1 // block location. This block will be the super-block BLOCK1
 Super-Block
 Block Name
 BLOCK1
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 6 // block location; this block formerly represented antiwindup compensation
 Dynamic Systems
 State-Space System
 Block Name
 WINDUP
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 0
 Parameter Entry
 EYE(NINPUTS)
//
 Define Block
 2 // block location. This block generates Ucm(t_i-1)
 Dynamic Systems
 Time Delay: exp(-kTs)
 Block Name
 DELAY
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
 DT // amount of time delay
//
 Define Block
 4 // block location. This block will be super-block GAIN1.
 Super-Block
 Block Name
 GAIN1
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 5 // block location. This block will be super-block KZYDM.
 Super-Block
 Block Name
 KZYTM
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 3 // block location. This block is the summing junction.
 Algebraic Equations
 Sum of Vectors
 Number of Outputs
 NINPUTS
 Parameter Entry
 4 // four input vectors
 1,1,-1,-1 // the sign of each respective input vector
//
 Top
 Mat
//
// End of LAW.BLOCKS0 command file
//
RETURN




// LAW.BLOCKS1 command file
//
```

```
//***************************************************************************
//
// This command file is part 2 of 6 which uses MATRIXX SYSTEM BUILD
//  commands to put together the incremental form CGT/PI control law
//   u(t_i) = u(t_i-1) - KX*[Xtm(t_i)-Xtm(t_i-1)] + KZ*Ycm(t_i-1) -
//            KZ*Ytm(t_i-1) + [KX*A11+A21]*[Xcm(t_i)-Xcm(t_i-1)]
//  which can also be seen in Maybeck's Volume III, equation 14-267.
//  The name of this command file represents the ouput of the super-
//  block, which is the control LAW, and that it generates the internal
//  BLOCKS of the super-block if the WINDUPFLAG=1 (antiwindup compensation
//  is turned on).
//
// The LAW super-block (which is generated by the LAW.* command files)
//  has an input of Ucm (the command model input) and an output of Udm
//  (the design model control law).  It uses the super-blocks GAIN1,
//  GAIN3, KZYCM, and KZYTM.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (1 Feb)
//***************************************************************************
//
 BUILD
 Edit Super-Block
 LAW // name of super-block
 DT // sample period
 0 // first sample time
//
 Define Block
 1 // block location. This block will be the super-block BLOCK1
 Super-Block
 Block Name
 BLOCK1
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 6 // block location. This block will be the antiwindup compensation
 Piece-Wise Linear
 L - U Bounded Limit
 Block Name
 WINDUP
 Number of Outputs
 NINPUTS
 Parameter Entry
 PLOWLIM
 PUPLIM
//
 Define Block
 2 // block location. This block generates Ucm(t_i-1)
 Dynamic Systems
 Time Delay: exp(-kTs)
 Block Name
 DELAY
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
 DT // amount of time delay
//
 Define Block
 4 // block location. This block will be super-block GAIN1.
 Super-Block
 Block Name
 GAIN1
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 5 // block location. This block will be super-block KZYDM.
 Super-Block
 Block Name
 KZYTM
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
```

```
//
 Define Block
 3 // block location. This block is the summing junction.
 Algebraic Equations
 Sum of Vectors
 Number of Outputs
 NINPUTS
 Parameter Entry
 4 // four input vectors
 1,1,-1,-1 // the sign of each respective input vector
//
 Top
 Mat
//
// End of LAW.BLOCKS1 command file
//
RETURN




// LAW.SUM command file
//
//**************************************************************************
//
// This command file is part 3 of 6 which uses MATRIXX SYSTEM BUILD
//  commands to put together the incremental form CGT/PI control law
//   u(t_i) = u(t_i-1) - KX*[Xdm(t_i)-Xdm(t_i-1)] + KZ*Ycm(t_i-1) -
//              KZ*Ydm(t_i-1) + [KX*A11+A21]*[Xcm(t_i)-Xcm(t_i-1)]
//  which can also be seen in Maybeck's Volume III, equation 14-267.
//  The name of this command file represents the output of the super-
//  block, which is the control LAW, and that it deals with the SUMming
//  junction.
//
// The LAW super-block (which is generated by the LAW.* command files)
//  has an input of Ucm (the command model input) and an output of Udm
//  (the design model control law).  It uses the super-blocks GAIN1,
//  BLOCK1 and KZYDM.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (15 Nov)
//************************************************************************
//
 BUILD
 Edit Super-Block
 LAW // name of super-block
 0 // same sample period as LAW.BLOCKS used when generating this super-block
// first sample time is not required when editing an existing super-block
//
 Connect Blocks // all blocks will be connected to block 3, the summing
junction
 Internal Path
 1
 3
 N,N // a connection
 0,0
 Internal Path
 2
 3
 N,(N+NINPUTS) // a connection
 0,0 // end connection mode
 Internal Path
 4
 3
 N,(N+2*NINPUTS) // a connection
 0,0
 Internal Path
 5
 3
 N,(N+3*NINPUTS) // a connection
 0,0
//
 Top
 Mat
//
// End of LAW.SUM command file
//
RETURN
```

```
// LAW.CONNECT2 command file
//
```

```
//*********************************************************************************
//
// This command file is part 4 of 6 which uses MATRIXX SYSTEM BUILD
//  commands to put together the incremental form CGT/PI control law
//   u(t_i) = u(t_i-1) - KX*[Xdm(t_i)-Xdm(t_i-1)] + KZ*Ycm(t_i-1) -
//             KZ*Ydm(t_i-1) + [KX*A11+A21]*[Xcm(t_i)-Xcm(t_i-1)]
//  which can also be seen in Maybeck's Volume III, equation 14-267.
//  The name of this command file represents the ouput of the super-
//  block, which is the control LAW, and that it deals with the CONNECTions
//  (both internal and external) with dimension of 2 or greater.  The
flag
//  is NINPUTS, because this is the dimension of the connections.
//
// The LAW super-block (which is generated by the LAW.* command files)
//  has an input of Ucm (the command model input) and an output of Udm
//  (the design model control law).  It uses the super-blocks GAIN1,
//  BLOCK1 and KZYDM.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (15 Feb)
//*********************************************************************************
//
 BUILD
 Edit Super-Block
 LAW // name of super-block
 0 // same sample period as LAW.BLOCKS used when generating this super-block
// first sample time is not required when editing an existing super-block
//
```

```
 Connect Blocks
 External Input
 NINPUTS // dimension of external input, which is the command model control.
 1 // external input goes to block 1.
 Y // connection is simple
//
// Establish the connections leading from the summing junction to blocks
6,
//  4, and 5, and from block 6 to 2.
//
 Internal Path
 6
 4
 Y // connection is simple
 Internal Path
 6
 5
 Y // connection is simple
 Internal Path
 3
 6
 Y // connection is simple
 Internal Path
 6
 2
 Y // connection is simple
//
 External Output
 NINPUTS
 6 // The ouput of the super-block is the output of block 6, antindup
//      compensation
 Y // connection is simple
//
 Top
 Mat
//
// End of LAW.CONNECT2 command file
//
RETURN




// LAW.CONNECT1 command file
//
//***************************************************************************
//
// This command file is part 6 of 6 which uses MATRIXX SYSTEM BUILD
//  commands to put together the incremental form CGT/PI control law
//   u(t_i) = u(t_i-1) - KX*[Xdm(t_i)-Xdm(t_i-1)] + KZ*Ycm(t_i-1) -
//              KZ*Ydm(t_i-1) + [KX*A11+A21]*[Xcm(t_i)-Xcm(t_i-1)]
//  which can also be seen in Maybeck's Volume III, equation 14-267.
//  The name of this command file represents the output of the super-
//  block, which is the control LAW, and that it deals with the CONNECTions
//  (both internal and external) with dimension of 1.  The flag is
//  NINPUTS, because this is the dimension of the connections.
```

```
//
// The LAW super-block (which is generated by the LAW.* command files)
//  has an input of Ucm (the command model input) and an output of Udm
//  (the design model control law).  It uses the super-blocks GAIN1,
//  GAIN3, KZYCM, and KZYDM.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (15 Feb)
//*****************************************************************************
//
 BUILD
 Edit Super-Block
 LAW // name of super-block
 0 // same sample period as LAW.BLOCKS used when generating this super-block
// first sample time is not required when editing an existing super-block
//
 Connect Blocks
 External Input
 NINPUTS // dimension of external input, which is the command model control.
 1 // external input goes to block 1.
//
// Establish the connections leading from the summing junction to blocks
6,
//  4, and 5, and block 6 to 2.
//
 Internal Path
 6
 4
 Internal Path
 6
 5
 Internal Path
 3
 6
 Internal Path
 6
 2
//
 External Output
 NINPUTS
 6 // The ouput of the super-block is the output of block 6 the antiwindup
//      compensation
//
 Top
 Mat
//
// End of LAW.CONNECT1 command file
//
RETURN
```

```
// ACT.BLOCKS0 command file
//
//*************************************************************************
//
// This command file is 1 of 4 which allows for actuator dynamics and
//  actuator limits to be included in the simulation.  It consists of
//  MATRIXX SYSTEM BUILD commands.
//
// The name of this command file is a mnemonic representing the fact
//  that this command file builds the blocks of super-block ACT if
//  the number of actuator states (NSACT) per actuator equals 0.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (1 Feb)
//*************************************************************************
//
 BUILD
 Edit Super-Block
 ACT // name of super-block
 0  // sample period (continuous system)
//
 Define Block
 1 // block location; this block is the LAW super-block
 Super-Block
 Block Name
 LAW
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 2 // block location; this block represents the actuator higher order
//     terms, that is, the terms above one derivative
 Dynamic Systems
 State-Space System
 Block Name
 ACTHOT
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 0 // If NSACT<2*ninputs, no higher order dynamics
 Parameter Entry
 SACTHOT
//
 Define Block
 3 // block location; this block represents rate limits
 Piece-Wise Linear
 L - U Bounded Limit
 Block Name
 RLIMIT
 Number of Outputs
 NINPUTS
 Parameter Entry
 RLOWLIM
 RUPLIM
//
 Define Block
 4 // block location; this block represents actuator first order terms
 Dynamic Systems
 State-Space System
 Block Name
 ACT1
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 0 // If NSACT=0, no actuator dynamics
 Parameter Entry
 SACT1
//
 Define Block
 5 // block location; this block represents position limits
 Piece-Wise Linear
 L - U Bounded Limit
 Block Name
 PLIMIT
 Number of Outputs
 NINPUTS
 Parameter Entry
 PLOWLIM
 PUPLIM
//
 TOP
 MAT
//
// End of ACT.BLOCKS0 command file
//
RETURN




// ACT.BLOCKS1 command file
//
//*****************************************************************************
//
// This command file is 2 of 4 which allows for actuator dynamics and
//  actuator limits to be included in the simulation.  It consists of
//  MATRIXX SYSTEM BUILD commands.
//
// The name of this command file is a mnemonic representing the fact
//  that this command file builds the blocks of super-block ACT if
//  the number of actuator states (NSACT) per actuator equals 1.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (1 Feb)
//*****************************************************************************
```

```
//
 BUILD
 Edit Super-Block
 ACT // name of super-block
 0  // sample period (continuous system)
//
 Define Block
 1 // block location
 SUPER-BLOCK
 Block Name
 LAW
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 2 // block location; this block represents the actuator higher order
//     terms, that is, the terms above one derivative
 Dynamic Systems
 State-Space System
 Block Name
 ACTHOT
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 0 // If NSACT<2*ninputs, no higher order dynamics
 Parameter Entry
 SACTHOT
//
 Define Block
 3 // block location; this block represents rate limits
 Piece-Wise Linear
 L - U Bounded Limit
 Block Name
 RLIMIT
 Number of Outputs
 NINPUTS
 Parameter Entry
 RLOWLIM
 RUPLIM
//
 Define Block
 4 // block location; this block represents actuator first order terms
 Dynamic Systems
 State-Space System
 Block Name
 ACT1
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 NINPUTS
 Parameter Entry
 SACT1
 N // initial states not zero
 XTMO((NSTM-NSACT+1):(NSTM-NSACT+NINPUTS))
// actuator higher order term state initial conditions
//
 Define Block
 5 // block location; this block represents position limits
 Piece-Wise Linear
 L - U Bounded Limit
 Block Name
 PLIMIT
 Number of Outputs
 NINPUTS
 Parameter Entry
 PLOWLIM
 PUPLIM
//
 TOP
 MAT
//
// End of ACT.BLOCKS1 command file
//
RETURN
```

```
// ACT.BLOCKS2 command file
//
//****************************************************************************
//
// This command file is 3 of 4 which allows for actuator dynamics and
//  actuator limits to be included in the simulation.  It consists of
//  MATRIXX SYSTEM BUILD commands.
//
// The name of this command file is a mnemonic representing the fact
//  that this command file builds the blocks of super-block ACT if
//  the number of actuator states (NSACT) per actuator is =>2.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (1 Feb)
//****************************************************************************
//
 BUILD
 Edit Super-Block
 ACT // name of super-block
 0  // sample period (continuous system)
//
 Define Block
 2 // block location
 Super-Block
 Block Name
 LAW
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 2 // block location; this block represents the actuator higher order
//     terms, that is, the terms above one derivative
 Dynamic Systems
 State-Space System
 Block Name
 ACTHOT
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 (NSACT-NINPUTS)
 Parameter Entry
 SACTHOT
 N // initial states not zero
 XTMO((NSTM-NSACT+NINPUTS+1):NSTM,:) // actuator higher order term state
//                                        initial conditions
//
 Define Block
 3 // block location; this block represents rate limits
 Piece-Wise Linear
 L - U Bounded Limit
```

```
Block Name
RLIMIT
Number of Outputs
NINPUTS
Parameter Entry
RLOWLIM
RUPLIM
//
Define Block
4 // block location; this block represents actuator first order terms
Dynamic Systems
State-Space System
Block Name
ACT1
Number of Inputs
NINPUTS
Number of Outputs
NINPUTS
Number of States
NINPUTS
Parameter Entry
SACT1
N // initial states not zero
XTMO((NSTM-NSACT+1):(NSTM-NSACT+NINPUTS))
// actuator higher order term state initial conditions
//
Define Block
5 // block location; this block represents position limits
```

```
 Piece-Wise Linear
 L - U Bounded Limit
 Block Name
 PLIMIT
 Number of Outputs
 NINPUTS
 Parameter Entry
 PLOWLIM
 PUPLIM
//
 TOP
 MAT
//
// End of ACT.BLOCKS2 command file
//
RETURN




// ACT.NOLIMITS command file
//
//****************************************************************************
//
// This command file is 4 of 4 which allows for actuator dynamics and
//  actuator limits to be included in the simulation.  It consists of
//  MATRIXX SYSTEM BUILD commands.
//
```

```
// The name of this command file is a mnemonic representing the fact
//  that this command file deletes the blocks of super-block ACT representing
//  the limits if the user chose not to use limits (LIMFLAG=0).
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (1 Feb)
//*****************************************************************************
//
 BUILD
 Edit Super-Block
 ACT // name of super-block
 0 // the sample period is same as defined in ACT.BLOCKS*
//
 Define Block
 3 // block location; this block formerly represented rate limits
 Dynamic Systems
 State-Space System
 Block Name
 RLIMIT
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 0
 Parameter Entry
 EYE(NINPUTS)
//
```

```
 Define Block
 5 // block location; this block formerly represented position limits
 Dynamic Systems
 State-Space System
 Block Name
 PLIMIT
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 0
 Parameter Entry
 EYE(NINPUTS)
//
 TOP
 MAT
//
// End of ACT.NOLIMITS command file
//
RETURN
```

```
// YTM.2 command file
//************************************************************************
//
// This command file generates the discrete output vector of the truth
//  model if the dimension of the connections is 2 or greater.  The flag
is
//  NINPUTS, because that is the dimension of the connections.  It
//  uses MATRIXX SYSTEM BUILD commands.
//
// Actuator dynamics are included in the ACT super-block, so these are
//  removed from the truth model defining matrices in block 3.
//  DUMMY is a dummy variable equal to NSTM-NSACT.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (13 Jan)
//************************************************************************
//
 BUILD
 Edit Super-Block
 YTM // name of super-block
 0 // sample period
//
 Define Block
 2 // block location
 Super-Block
 Block Name
 ACT // this super-block generates the actuator dynamics and limits
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 3 // block location. This block is the truth model
 Dynamic Systems
 State-Space System
 Block Name
 STM
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS // number of truth model outputs
 Number of States
 DUMMY
 Parameter Entry
 SY // defined in CONTROL
 N // initial states not zero
 XTMO(1:DUMMY) // truth model initial conditions
//
 Connect Blocks
 Internal Path
 2
 3
 Y // simple connection
 External Input
 NINPUTS // number of inputs
 2 // block location
 Y // simple connection
 External Output
 NINPUTS // number of outputs
 3 // block location
 Y // simple connection
 Describe Blocks
 TOP
 MAT
//
// End of YTM.2 command file
//
RETURN




// YTM.1 command file
//***********************************************************************
//
// This command file generates the discrete output vector of the truth
//  model if the dimension of the connections is 1.  The flag is
//  NINPUTS, because that is the dimension of the connections.  It
//  uses MATRIXX SYSTEM BUILD commands.
//
// Actuator dynamics are included in the ACT super-block, so these are
//  removed from the truth model defining matrices in block 3.
//  DUMMY is a dummy variable equal to NSTM-NSACT.
//
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (1 Feb)
//***********************************************************************
//
 BUILD
 Edit Super-Block
 YTM // name of super-block
 0 // sample period
//
 Define Block
 2 // block location
 Super-Block
 Block Name
 ACT // this super-block generates the actuator dynamics and limits
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 3 // block location.  This block represents the truth model
 Dynamic Systems
 State-Space System
 Block Name
 STM
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 DUMMY
 Parameter Entry
 SY
 N // initial states not zero
 XTMO(1:DUMMY) // truth model initial conditions
//
 Connect Blocks
 Internal Path
 2
 3
//
 External Input
 NINPUTS // number of inputs
 2 // block location
 External Output
 NINPUTS // number of outputs
 3 // block location
 Describe Blocks
 TOP
 MAT
//
// End of YTM.1 command file
```

```
//

RETURN
```

```
// XTM.2 command file
//************************************************************************
//
// This command file generates the controlled discrete states of the truth
//  model if the dimension of the connections is 2 or greater.  The flag
is
//  NINPUTS, because that is the dimension of the connections.  It
//  uses MATRIXX SYSTEM BUILD commands.
//
// This command file is called by the CONTROL command file.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (1 Feb)
//************************************************************************
//
 BUILD
 Edit Super-Block
 XTM // name of super-block
 0 // sample period
//
 Define Block
 2 // block location
 Super-Block
 Block Name
 ACT // this super-block generates the actuator dynamics and limits
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 3 // block location. This block is the truth model
 Dynamic Systems
 State-Space System
 Block Name
 TMCONT
 Number of Inputs
 NINPUTS
 Number of Outputs
 DUMMY // number of truth model states less actuator states
 Number of States
 DUMMY
 Parameter Entry
 SX // defined in CONTROL command file
 N // initial states not zero
 XTMO(1:DUMMY) // truth model initial conditions
//
 Connect Blocks
 Internal Path
 2
 3
 Y // simple connection
 External Input
 NINPUTS // number of inputs
 2 // block location
 Y // simple connection
 External Output
 DUMMY // number of outputs
 3 // block location
 Y // simple connection
 Describe Blocks
 TOP
 MAT
//
// End of XTM.2 command file
//
RETURN




// XTM.1 command file
//*************************************************************************
//
// This command file generates the controlled discrete states of the truth
//  model if the dimension of the connections is 2 or greater.  The flag
is
//  NINPUTS, because that is the dimension of the connections.  It
//  uses MATRIXX SYSTEM BUILD commands.
//
// This command file is called by the CONTROL command file.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (1 Feb)
//************************************************************************
//
 BUILD
 Edit Super-Block
 XTM // name of super-block
 0 // sample period
//
 Define Block
 2 // block location
 Super-Block
 Block Name
 ACT // this super-block generates the actuator dynamics and limits
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 3 // block location. This block is the truth model
 Dynamic Systems
 State-Space System
 Block Name
 TMCONT
 Number of Inputs
 NINPUTS
 Number of Outputs
 DUMMY // number of truth model states less actuator states
 Number of States
 DUMMY
 Parameter Entry
 SX
 N // initial states not zero
 XTMO(1:DUMMY) // truth model initial conditions
//
 Connect Blocks
 Internal Path
 2
 3
 External Input
 NINPUTS // number of inputs
 2 // block location
 External Output
 DUMMY // number of outputs
 3 // block location
 Y // conection is simple; case of 1 truth model state not covered
 Describe Blocks
 TOP
 MAT
//
// End of XTM.1 command file
//
RETURN
```

```
// YCMD.2 command file
//*************************************************************************
//
// This command file is part 2 of 2 which generates the command model
//  output (and the YCMD super-block) for use by the  CGTPI.ANALYZE
//  command file.  It is called by the CONTROL command file, but it
//  is not used in the generation of the feedback control law.  The
//  first part of the command model generation program is called if the
//  external connections are equal to 1, while this part of the
//  program is called if the connections are greater than 1.  NINPUTS
is the
//  flag, since that is the dimension of the connections.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (16 Dec)
//*************************************************************************
//
 BUILD
 Command // change menu
 Set Auto PLOT OFF // turn graphics off
 Top // change menu
 Edit Super-Block
 YCMD // name of super-block
 DT // sample period
 0 // first sample time
//
 Define Block
 1 // block loation
 Dynamic Systems
 State-Space System
 Block Name
 COMMOD // this block is the state space matrix [a,b;c,d]
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 NSCM
 Parameter Entry
 [ACMD,BCMD;CCM,DCM]
 N // initial states not zero
 XCMO // command model initial conditions
//
// Connect the blocks
//
 Connect Blocks
 External Input
 NINPUTS // number of inputs to the super-block
 1 // external inputs come into block 1
 Y // connection is simple
 External Output
 NINPUTS // number of outputs from the super-block
 1 // external outputs come from block 1
 Y // connection is simple
//
 TOP
 MAT
//
// End of YCMD.2 command file
//
RETURN
```

```
// YCMD.1 command file
//***************************************************************************
//
// This command file is part 1 of 2 which generates the command model
//  output (and the YCMD super-block) for use by the  CGTPI.ANALYZE
//  command file.  It is called by the CONTROL command file, but it
//  is not used in the generation of the feedback control law.  The
//  second part of the command model generation program is called if the
//  external connections are greater than 1, while this part of the
//  program is called if the connections are equal to 1.  NINPUTS is the
//  flag, since that is the dimension of the connections.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (16 Dec)
//***************************************************************************
//
 BUILD
 Command // change menu
 Set Auto PLOT OFF // turn graphics off
 Top // change menu
 Edit Super-Block
 YCMD // name of super-block
 DT // sample period
 0 // first sample time
//
 Define Block
 1 // block loation
 Dynamic Systems
 State-Space System
 Block Name
 COMMOD // this block is the state space matrix [a,b;c,d]
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 NSCM
 Parameter Entry
 [ACMD,BCMD;CCM,DCM]
 N // initial states not zero
 XCMO // command model initial conditions
//
// Connect the blocks
//
 Connect Blocks
 External Input
 NINPUTS // number of inputs to the super-block
 1 // external inputs come into block 1
 External Output
 NINPUTS // number of outputs from the super-block
 1 // external outputs come from block 1
//
 TOP
 MAT
//
// End of YCMD.1 command file
//
RETURN
```

B.4 KF and Associated Sub-files

```
// KF command file
//
//**************************************************************************
//
// This command file deals with the Kalman filter.  It allows the user
//  to input the noise matrices, generate the Kalman filter gain and
//  covariance matrices, and evaluate the filter against a truth model.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (9 Jan)
//**************************************************************************
//
// Define a menu for the appropriate options
//
KFMENU = [' KALMAN FILTER MENU '
          ' MODIFY G,Q,R,OR QR '
          ' GENERATE GAINS,COV '
          ' COVARIANCE ANALYSIS'
          '       QUIT         '];
//
// Now use the menu (Z is the option parameter)
//
Z=0;
WHILE Z<>4,...
 Z=MENU(KFMENU);...
...//
 IF Z=1, EXECUTE('INPUT.KF'),END,...
```

```
...//
 IF Z=2;...
   [SDMD]=DISCRETIZE(SDM,NSDM,DT);...//Generate ADMD
   [ADMD,BDMD,CDMD,DDMD]=SPLIT(SDMD,NSDM);...
   QD1=ADMD*GDM*QDM*GDM'*ADMD';...// Note that QDMD incorporates the G
term
   QDMD=(QD1+GDM*QDM*GDM')*.5*DT;...//From Maybeck Vol I, eq 6-118
   [KFEVAL,KFGAIN,PFMINUS]=DESTIMATOR(ADMD,HDM,QDMD,RDM,QRDM);...
...//
...//KFGAIN as generated by DESTIMATOR is a composite of measurement
...// and update equation. This next step changes it to just measurement.
...//
   KFGAIN=INV(ADMD)*KFGAIN;...
   DISP('THE FILTER DISCRETE EIGENVALUES, GAIN, AND COVARIANCE MATRICES'),...
   DISP('(PFMINUS AND PFPLUS RELATE TO BEFORE OR AFTER MEASUREMENT'),...
   DISP('UPDATE) ARE:'),...
   KFEVAL,PAUSE,KFGAIN,PAUSE,PFMINUS,PAUSE,...
   PFPLUS = PFMINUS - KFGAIN*HDM*PFMINUS,PAUSE,...
 END,...
...//
...// Perform filter analysis
...//
 IF Z=3,...
...//
...// This next command file has a lot of variables internal to it, so
...//   we'll save some to prevent overloading the buffer.
...//
  FSAVE 'GARBAGE' ACM ACMD BCM BCMD CCM DCM SCM SCMD EX EY ATM ...
```

```
    BTM CTM DTM ADM BDM DDM ANM GNM ADM BDM ,...
  CLEAR ACM ACMD BCM BCMD CCM DCM SCM SCMD EX EY ATM ...
    BTM CTM DTM ADM BDM DDM ANM GNM ADM BDM ,...
  EXECUTE('COV.ANALYSIS'),...
  LOAD 'GARBAGE',...
 END,...
...//
END
//
// Clear extraneous variables
//
CLEAR Z QD1 KFMENU
//
// End of KF command file
//
RETURN
```

```
// INPUT.KF command file
//
//**********************************************************************
//
// This command file allows the user to input the Kalman filter
//  noise strength matrices.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (9 Jan)
//**********************************************************************
//
DISP('YOU CAN NOW INPUT THE KALMAN FILTER NOISE STRENGTH MATRICES.')
DISP('THESE ARE GDM, A CONTINUOUS MATRIX WHICH DETERMINES HOW THE')
DISP('DRIVING NOISE ENTERS THE SYSTEM; QDM, WHICH IS THE CONTINUOUS')
DISP('SYSTEM DYNAMICS DRIVING NOISE; RDM, WHICH IS THE DISCRETE')
DISP('MEASUREMENT NOISE; AND QRDM, WHICH IS THE DISCRETE CROSS')
DISP('CORRELATION BETWEEN QDM AND RDM.')
DISP(' ')
PAUSE
DISP(' ')
DISP('ALL OF THESE NOISES ARE ASSUMED TO BE ZERO-MEAN WHITE GAUSSIAN')
DISP('SEQUENCES. THE DM AT THE END OF EACH MATRIX DESIGNATES IT AS')
DISP('BEING ASSOCIATED WITH THE DESIGN MODEL, DM.')
DISP(' ')
PAUSE,DISP(' ')
//
// Input the GDM matrix, and perform syntax checks
//
```

```
DUMMY=0;
WHILE DUMMY<1,...
 DISP('GDM MUST HAVE THE SAME NUMBER OF ROWS AS THE DESIGN MODEL.'),NSDM,...
 DISP('GDM ALREADY EXISTS, SO IF YOU DONT WANT TO CHANGE IT, JUST'),...
 DISP('ENTER GDM.'),...
 INQUIRE GDM ' GDM = ',...
 [ROWS,COLUMNS]=SIZE(GDM);IF ROWS<>NSDM,...
   DISP('WRONG NUMBER OF ROWS. TRY AGAIN.'),...
   ELSE DUMMY=1;END,...
END
//
// Input the QDM matrix, and perform syntax checks
//
DUMMY=0;
WHILE DUMMY<1,...
 DISP('QDM MUST BE SQUARE WITH DIMENSIONS OF THE NUMBER OF COLUMNS'),...
 DISP('OF THE GDM MATRIX, AND ALSO BE POSITIVE SEMI-DEFINITE.'),COLUMNS,...
 INQUIRE QDM 'QDM = ',...
 [ROWS,COL]=SIZE(QDM);...
 IF ROWS<>COLUMNS, DISP('WRONG NUMBER OF ROWS. TRY AGAIN'),...
   ELSE IF COL<>COLUMNS,...
     DISP('WRONG NUMBER OF COLUMNS. TRY AGAIN.'),...
     ELSE DUMMY=1;QEVAL=EIG(QDM);... // Test for semi-definiteness
     FOR ZZ=1:COLUMNS,...
       ZEVAL=QEVAL(ZZ,1);...
       IF ZEVAL<0, DISP('MATRIX MUST BE POSITIVE SEMI-DEFINITE. TRY AGAIN.'),
         DUMMY=-NSDM;END,...
     END,...
```

```
...// Test for symmetry
     FOR I=1:COLUMNS,...
       FOR J=1:COLUMNS,...
         IF QDM(I,J)<>QDM(J,I),DISP('QDM MUST BE SYMMETRIC. TRY AGAIN.'),...
           DUMMY=-NSDM;END,...
     END,END,...
 END,END,...
END
//
// Input the RDM matrix, and perform syntax checks
//
DUMMY=0;
WHILE DUMMY<1,...
 DISP('RDM MUST BE SQUARE WITH DIMENSIONS OF THE DESIGN MODEL'),...
 DISP('MEASUREMENTS'),NMEAS,...
 DISP('AND ALSO BE POSITIVE DEFINITE AND SYMMETRIC. IT IS ASSUMED TO'),...
 DISP('REPRESENT DISCRETE TIME MEASUREMENTS, AND WILL NOT'),...
 DISP('DISCRETIZED LATER.'),...
 INQUIRE RDM 'RDM = ',...
 [ROWS,COL]=SIZE(RDM);...
 IF ROWS<>NMEAS, DISP('WRONG NUMBER OF ROWS. TRY AGAIN'),...
   ELSE IF COL<>NMEAS,...
     DISP('WRONG NUMBER OF COLUMNS. TRY AGAIN.'),...
     ELSE DUMMY=1;REVAL=EIG(RDM);... // Test for definiteness
     FOR ZZ=1:NMEAS,...
       ZEVAL=REVAL(ZZ,1);...
       IF ZEVAL<=0, DISP('MATRIX MUST BE POSITIVE DEFINITE. TRY AGAIN.'),...
         DUMMY=-NMEAS;END,...
     END,...
...// Test for symmetry
     FOR I=1:NMEAS,...
       FOR J=1:NMEAS,...
         IF RDM(I,J)<>RDM(J,I),DISP('RDM MUST BE SYMMETRIC. TRY AGAIN.'),...
           DUMMY=-MEAS;END,...
     END,END,...
 END,END,...
END
//
// Input the cross-correlation matrix, QRDM, and perform syntax checks
//
DISP(' ')
DUMMY=0;
WHILE DUMMY<1,...
 DISP('QRDM MUST HAVE DIMENSIONS '),ROWS=NMEAS,COLUMNS=NSDM,...
 PAUSE,...
 DISP('YOU MAY INPUT A SINGLE ZERO IF YOU DO NOT WANT TO HAVE'),...
 DISP('CROSS CORRELATION TERMS. THIS IS ASSUMED TO BE A DISCRETE'),...
 DISP('TIME MATRIX.'),...
 INQUIRE QRDM 'QRDM = ',...
...//
...// If the user inputs a zero, make QRDM the correct dimensions
...//
 [ROWS,COL]=SIZE(QRDM); IF ROWS=1, IF COL=1,...
     IF QRDM=0,QRDM=0*ONES(NMEAS,NSDM), END,END,END,...
...//
...// Ensure QRDM has the correct dimensions.  If a 0 was input, it already
```

```
...//  will, but we'll check anyway.
...//
 [ROWS,COL]=SIZE(QRDM);...
 IF ROWS<>NMEAS, DISP('THE NUMBER OF ROWS WAS WRONG. TRY AGAIN.'),...
   ELSE IF COL=NSDM, DUMMY=1; ELSE...
     DISP('THE NUMBER OF COLUMNS WAS WRONG. TRY AGAIN.'),...
 END,END,...
END
//
// Display the matrices to the user
//
DISP('THE NOISE MATRICES ARE'),GDM,PAUSE,QDM,PAUSE,RDM,PAUSE,QRDM,PAUSE
//
CLEAR I J DUMMY ROWS COL COLUMNS ZZ ZEVAL// Clear extraneous variables
CLEAR QEVAL REVAL
//
// End of INPUT.KF command file
//
RETURN
```

```
// COV.ANALYSIS command file
//
//**************************************************************************
//
// This command file is based on the KFEVAL program written by Dr. Peter
S.
//  Maybeck, and modified by Captain Charles Sokol, AFIT GE/88D.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (20 Jan)
//**************************************************************************
//
// Form Augmented Matrices for Performance Evaluation
// ---------------------------------------------------
//
//
[STMD] = DISCRETIZE(STM,NSTM,DT); // Form discretized truth model matrices
[ATMD,BTMD,CTM,DTM] = SPLIT(STMD,NSTM);//CTM and DTM don't change
[I,J]=SIZE(GTM);
GAUG = [GTM;0*ONES(NSDM,J)];//GDM has dimensions of NSDM, GTM of NSTM
//
//ASSUME (FOR NOW) THAT XT AND X DO NOT EXIST
//IF THEY DID, AAUGD= [ATMD,-XT;0,ADMD-X] , IF X AND XT ARE DISCRETE (THEY'RE
// NOT, WE'D HAVE TO FORM AAUG AND DISCRETIZE)
//
AAUGD = [ATMD, 0*ONES(NSTM,NSDM); 0*ONES(NSDM,NSTM), ADMD];
QAUGD1 = AAUGD*GAUG*QTM*GAUG'*AAUGD';
QAUGD = (QAUGD1 + GAUG*QTM*GAUG')*.5*DT;//Maybeck Vol I, eq 6-118
```

```
CLEAR QAUGD1 I J STMD
//
//ASSUME THAT DT AND D DO NOT EXIST. IF THEY DID,
//DAUG = [EYE(NSTM), -DT; 0*ONES(NSDM,NSTM), EYE(NSDM)-D];
// I'D HAVE TO CHANGE THE NAME OF DT, SO IT WOULDN'T BE SAMPLE TIME
//
DAUG = [EYE(NSTM), 0*EYE(NSDM); 0*ONES(NSDM,NSTM), EYE(NSDM)];
CAUG = [-CTM, CDM]; // CTM AND CDM SAME NUMBER OF ROWS? CDM=I?
PAUGO = [PTO, 0*ONES(NSTM,NSDM); 0*ONES(NSDM,NSTM), 0*ONES(NSDM)];
//IF DETERMINISTIC TM, PAUGO=0
//
// KFGAIN has dimensions NSDM x NMEAS; HDM has dim NMEAS x NSDM
//  HTM has dimensions TMNMEAS x NSTM
AA = [EYE(NSTM), 0*ONES(NSTM,NSDM); KFGAIN*HTM, EYE(NSDM)-KFGAIN*HDM];
KAUG = [0*ONES(NSTM,NINPUTS); KFGAIN]; //K (Kalman filter gain) augmented
//
//*****************************************************************
//
// CONDUCT THE PERFORMANCE ANALYSIS
// -------------------------------
//
// Over [ITOTAL] sample periods, perform propagation, measurements
// update, impulsive reset, and error covariance computations
//
//*****************************************************************
//
// Initialize
// ----------
// PAM = P (covariance matrix) augmented before update (t-) (minus)
// PEM = P error minus (t-)
// PEMF = P of filter before update (t -)
// PEPF = P error of filter after update (t+)
// PEPCF = P error of filter after update (t+) and impulsive Control update
// PEPC = augmented P error after update (t+) and impulsive Control update
//
// (by 'P' I mean the covariance matrix)
//
PAM = PAUGO;...
PEM = CAUG * PAM * CAUG';...//augmented P (t -) error
PEMF = CDM * PFMINUS * CDM';...//P error at t- of filter
PAP = AA * PAM * AA' + KAUG * RTM * KAUG';
PEP = CAUG * PAP * CAUG';
PEPF = CDM * PFPLUS * CDM';
PAPC = DAUG * PAP * DAUG';
PEPC = CAUG * PAPC * CAUG';
PEPCF = PEPF;
//
//
// Form matrices with rows as DIAGonals from Pe(-,+,+c) from each
// sample time:
//
//
//  Pem(1,1)   Pem(2,2)   ...   Pem(p,p)
//  Pep(1,1)   Pep(2,2)   ...   Pep(p,p)      }   for time to
// Pepc(1,1)  Pepc(2,2)   ...  Pepc(p,p)
// --------------------------------------
//  Pem(1,1)    Pem(2,2)    ...    Pem(p,p)
//  Pep(1,1)    Pep(2,2)    ...    Pep(p,p)       }   for time t1
// Pepc(1,1)   Pepc(2,2)    ...   Pepc(p,p)
// ---------------------------------------
//
// and analogous partitions for t2 to end
//
// and similarly for filter-computed Pef(-,+,+c)
//
// Initialize the following variables.  The square root of main
//  diagonals of the covariance matrices (P) will give standard.
//  deviations.  That is the purpose of EM, EP, EPC, EMF, EPF,
//  and EPCF.
//
EM = DIAG(PEM);
EP = DIAG(PEP);
EPC = DIAG(PEPC);
//
ETRUE = [em'; ep'; epc'];// Column vectors result from the DIAG command
//
EMF = DIAG(PEMF);
EPF = DIAG(PEPF);
EPCF = DIAG(PEPCF);
//
EFILT = [EMF'; EPF'; EPCF'];...
//
//******************************************************************
//
```

```
// Main Loop:  Iterate for i = 1 to i = ITOTAL
// ---------
//
INQUIRE TIME 'ENTER TIME DURATION OF COVARIANCE RUN, IN SECONDS '
ITOTAL=TIME/DT;
DISP('THIS MAY TAKE A WHILE. PLEASE BE PATIENT.')
FOR COUNT = 1:ITOTAL;...
...// Propagate to t-
  PAM = AAUGD * PAPC * AAUGD' + QAUGD;...
  PEM = CAUG * PAM * CAUG';...
  PEMF = CDM * PFMINUS * CDM';...
...// Perform measurement update at t+
  PAP = AA * PAM * AA' + KAUG * RTM * KAUG';...
  PEP = CAUG * PAP * CAUG';...
  PEPF = CDM * PFPLUS * CDM';...
...// Perform impulsive control update (c) after t+
  PAPC = DAUG * PAP * DAUG';...
  PEPC = CAUG * PAPC * CAUG';...
  PEPCF = PEPF;...
...//
...// Square root of main diagonals of covariance matrices (P) will
...//  give standard deviations.  Take out main diagonal now.
...//
  EM = DIAG(PEM);...
  EP = DIAG(PEP);...
  EPC = DIAG(PEPC);...
  ETRUE = [ETRUE; EM'; EP'; EPC'];...
  EMF = DIAG(PEMF);...
  EPF = DIAG(PEPF);...
  EPCF = DIAG(PEPCF);...
  EFILT = [EFILT; EMF'; EPF'; EPCF'];...
END;
//
// End of Main Loop to Conduct Performance Analysis
// ------------------------------------------------
//
// Clear extraneous variables
//
CLEAR EM EP EPC EMF EPF EPCF PAM PEM PEMF PAP PEP PEPF PAPC
CLEAR PEPC PEPCF AA KAUG PAUGO GAUG AAUGD QAUGD DAUG CAUG COUNT
//
//******************************************************************
//
// Establish Data Files for Plotting
// ---------------------------------
//
// Take square roots of all variances
//
RTETRUE = sqrt(ETRUE);
RTEFILT = sqrt(EFILT);
//
// Form vector of times for plotting "-", "+" and "+c" values:
// [0 0 0 1 1 1 ... ITOTAL ITOTAL ITOTAL]
//
TIME = [0 0 0]; // Initialize time
FOR I = 1:ITOTAL;...
```

```
  TIME = [TIME,I,I,I];...
END
TIME = DT * TIME';
//
// Now plot the data
//
DISP(' ')
DISP('DATA IS NOW READY FOR PLOTTING.  A PLOT WILL BE GENERATED FOR EACH')
DISP('SEPERATE STATE OF THE TRUE VS FILTER COMPUTED COVARIANCES.')
DISP('FOR OPTION 1, THE TRUE ERROR WILL BE PLOTTED FIRST, THEN THE')
DISP('FILTER VALUES.')
PAUSE
PLOTFLAG=0;// PLOTFLAG is a flag variable used to set the plotting mode
//             in the PLOT.HARDCOPY command file.
//
// Define a menu of plotting options
//
PLOTMENU= [' PLOTTING OPTIONS  '
           'TRUE VS FILTER COV '
           'TRUE COV +- 1 SIGMA'
           'FILTER COV +- 1 SIG'
           '      QUIT        '];
//
// Now use the menu
//
Z=0;
WHILE Z<>4,...
 Z=MENU(PLOTMENU);...
```

```
...//
 IF Z=1,...
   [I,J]=SIZE(RTETRUE);...// Need J
   FOR I=1:J,...
     STATE=[ETRUE(:,J),EFILT(:,J)];...
     PLOT(TIME,STATE,'YLABEL/TRUE VS FILTER COV/'),...
     INQUIRE ZZ 'ENTER 0 IF YOU WANT A HARDCOPY, ELSE ENTER 1 ',...
     IF ZZ=0, EXECUTE('PLOT.HARDCOPY'),END,...
   END,...
 END,...
...//
 IF Z=2,...
   [I,J]=SIZE(RTETRUE);...
   FOR I=1:J,...
     STATE=[ETRUE(:,J),ETRUE(:,J)+RTETRUE(:,J),ETRUE(:,J)-RTETRUE(:,J)];...
     PLOT(TIME,STATE,'YLABEL/TRUE COV+-1 SIGMA/'),...
     INQUIRE ZZ 'ENTER 0 IF YOU WANT A HARDCOPY, ELSE ENTER 1 ',...
     IF ZZ=0, EXECUTE('PLOT.HARDCOPY'),END,...
   END,...
 END,...
...//
 IF Z=3,...
   [I,J]=SIZE(RTEFILT);...
   FOR I=1:J,...
     STATE=[EFILT(:,J),EFILT(:,J)+RTEFILT(:,J),EFILT(:,J)-RTEFILT(:,J)];...
     PLOT(TIME,STATE,'YLABEL/FILTER COV+-1 SIG/'),...
     INQUIRE ZZ 'ENTER 0 IF YOU WANT A HARDCOPY, ELSE ENTER 1 ',...
     IF ZZ=0, EXECUTE('PLOT.HARDCOPY'),END,...
```

```
    END,...
   END,...
...//
ERASE,...// Clear the screen
END
Z=0;// Z is also used by the KF command file, which called COV.ANALYSIS
//
// End of plotting options
//
CLEAR ITOTAL I J STATE PLOTMENU PLOTFLAG ZZ EFILT ETRUE RTEFILT RTETRUE
//
// End of COV.ANALYSIS command file
//
RETURN
```

B.5 CGTPIKF.ANALYZE and Associated Sub-files

```
// CGTPIKF.ANALYZE command file
//
//***************************************************************************
//
// This command file allows the user to analyze the closed-loop CGT/PI
//  control system with Kalman filter in the loop. Limitations
//  exist on how long a MATRIXX line can be (4096 characters), so instead
//  of having one menu, the different options are divided into more menus.
//  One menu would effectively turn this entire command file into one
//  line of code, which results in the 'input buffer' overflowing.
//
// This command file calls the PLOTHELP function, which is a subroutine
//  incorporating commands used in plotting options.  PLOTHELP generates
//  the TIME and STRENGTH vectors.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (28 Jan)
//***************************************************************************
//
DEFINE 'PLOTHELP.'; // User defined function
DEFINE 'PLOTLOOP.'; // User defined function
ERASE // Clear the screen
DISP('THIS PART OF THE PROGRAM ALLOWS YOU TO GET TIME RESPONSE PLOTS')
DISP('OF THE CLOSED LOOP CGT/PI WITH KALMAN FILTER IN THE LOOP.')
DISP('THE FIRST MENU ALLOWS YOU TO PURSUE LTR TUNEING, AND TO LOAD NEW
FILES')
DISP('SUCH AS A NEW TRUTH MODEL.  THE TRUTH MODEL MUST EXIST BEFORE YOU')
DISP('CONTINUE.  ANYTIME YOU MODIFY ANY ASPECT OF THE SYSTEM (EG, GAINS),')
DISP('YOU MUST ENTER OPTION 3, BUILD THE SYSTEM, BEFORE PURSUING ANY ANALYSIS
DISP('THE SECOND (NEXT) MENU ALLOWS YOU TO GET THE TIME RESPONSE PLOTS.')
PAUSE
//
// Define a menu for the appropriate functions
//
ANMENU= ['CGTPIKF.ANLYZ MENU1'
         'PERFORM LTR TUNING '
         '   LOAD A FILE     '
         ' BUILD THE SYSTEM  '
         '    NEXT MENU      '];
//
// Now use the menu (Z is the option parameter)
//
Z=0;
WHILE Z<>4,...
 Z=MENU(ANMENU);...
...//
...// Option 1 allows the user to pursue loop transfer recovery
...//
 IF Z=1, EXECUTE('LTR.'), END,...
...//
...// Option 2 allows the user to input a new file, such as a new truth
...//   model to compare the controller against.
...//
 IF Z=2, EXECUTE('INPUTFILE.'),...
   DISP('IF YOU LOADED A NEW FILE THAT YOU WANT TO COMPARE THE CONTROLLER'),.
   DISP('AGAINST, YOU NEED TO PURSUE OPTION 3, BUILD THE SYSTEM, BEFORE'),...
   DISP('PURSUING ANY ANALYSIS.'),PAUSE,...
 END,...
...//
...// Option 3 builds the system.  It calls the command file CONTROL.
...//
 IF Z=3,  EXECUTE('CONTROL.WITHKF'),END,...
...//
END
//
DISP('NOW YOU CAN GET TIME RESPONSE PLOTS.  DES MOD IS THE UNCONTROLLED
')
DISP('DESIGN MODEL.  COM MOD IS THE COMMAND MODEL.  TRUTH MODEL OUTPUTS')
DISP('ARE THE TRUTH MODELS OUTPUTS WITH THE CONTROLLER IN THE LOOP.')
DISP('ACTUATOR RESPONSE IS THE RESPONSE OF THE ACTUATORS.  IF ACTUATORS
ARE')
DISP('ZERO ORDER, ACTUATOR RESPONSE IS THE OPTIMAL CONTROL GENERATED BY
THE')
DISP('CONTROLLER, AND IS THE INPUT TO THE TRUTH MODEL BEING EVALUATED.')
PAUSE
//
// Start the next menu
//

ANMENU= ['CGTPIKF.ANLYZ MENU2'
         'PLOT DES MOD OUTPUT'
         'PLOT COM MOD OUTPUT'
         'TRUTH MODEL OUTPUTS'
```

```
          'TRUTH MODEL STATES '
          ' LOOK AT VARIABLES '
          ' ACTUATOR RESPONSE '
          '     QUIT         '];

//
// Now use the menu (Z is the option parameter). PL is a flag for plot
//  hardcopies, which affects how the plot options (slower but prettier).
//
Z=0;PL=0;
WHILE Z<>7,...
 Z=MENU(ANMENU);...
...//
...// Option 1 allows the user to plot the uncontrolled design model response
...//  YDMD.
...//
 IF Z=1,...
   DISP('ENTER TIME DURATION'),...
   INQUIRE TIME 'TIME IN SECONDS = ',...
   NPTS=TIME/DT + 1;...
   [N,YDMD1]=DSTEP(SDMD,NSDM,NPTS);...
...//
...// MATRIXX generates outputs responses vs each input.  Obtaining a
...//  consolidated output makes more sense.
...//
   YDMD=0*YDMD1;... // Initialize YDMD
   FOR I=1:NINPUTS,...
    FOR J=1:NINPUTS,...
```

```
     YDMD(:,I)=YDMD(:,I)+YDMD1(:,J);...
   END,END,...
...//
   TIME=N*DT;...
   [PL]=PLOTLOOP(TIME,YDMD,PL);...
   PAUSE,...
   ERASE,... // clear the screen
   CLEAR YDMD1 N ,...
 END,...
...//
...// Option 2 allows the user to plot the command model ideal response,
...//  YCMD.
...//
 IF Z=2,...
   DISP('THE PROGRAM WILL NOW GENERATE THE COMMAND MODEL RESPONSE'),...
   DISP('(YCMD) TO A STEP INPUT'),...
   [TIME,UCM]=PLOTHELP(SCMD,NSCM,NINPUTS,DT,1);... // user defined function
   EXECUTE('ANALYZE.YCMD'),...// This calls a SYSTEM BUILD command file
   YCMD=SIM(TIME,UCM);...// This simulates the com model with UCM as the
input
   PL=PLOTLOOP(TIME,YCMD,PL);...
   ERASE,... // Clear the screen
   END,...
...//
...// Option 3 allows the user to view the controlled system's output
response,
...//  YCONT.
...//
```

```
 IF Z=3,...
   DISP('THE PROGRAM WILL NOW GENERATE THE CONTROLLED SYSTEM RESPONSE'),...
   DISP('OF THE TRUTH MODEL (YTM) TO A STEP INPUT'),...
   [TIME,UCM]=PLOTHELP(STMD,NSTM,NINPUTS,DT,1);...
   EXECUTE('ANALYZE.YTMF'),...// This calls a SYSTEM BUILD command file
   YTM=SIM(TIME,UCM);...// This simulates the system with UCM as the input
   PL=PLOTLOOP(TIME,YTM,PL);...
 END,...
...//
...// Option 4 lets the user see the controlled system's state response,
...//  XCONT.
...//
 IF Z=4,...
   DISP('THE PROGRAM WILL NOW GENERATE THE CONTROLLED TRUTH MODEL'),...
   DISP('STATE RESPONSE (XTM) TO A STEP INPUT'),...
   [TIME,UCM]=PLOTHELP(SDMD,NSDM,NINPUTS,DT,1);...
   EXECUTE('ANALYZE.XTMF'),...// This calls a SYSTEM BUILD command file
   XTM=SIM(TIME,UCM);...// This simulates the system with UCM as the input
   PLOTLOOP(TIME,XTM);...
 END,...
...//
IF Z=5,INQUIRE X 'ENTER VARIABLE YOU WANT TO SEE (X): ',X,PAUSE,...
CLEAR X ,...
END,...
...//
IF Z=6,...
 DISP('THE PROGRAM WILL NOW PLOT THE ACTUATOR RESPONSE THAT RESULTS'),...
 DISP('FROM THE DESIGN EFFORT.'),...
 [TIME,UCM]=PLOTHELP(SDMD,NSDM,NINPUTS,DT,1);...
 EXECUTE('ANALYZE.ACTF'),... // This calls a SYSTEM BUILD command file
 ACTUATOR=SIM(TIME,UCM);...
 DISP('ENTER THE TITLE, WITH QUOTATION MARKS AT BEGINNING AND END.'),...
 INQUIRE TITLE 'ENTER TITLE',...
 PLOT(TIME,ACTUATOR,['TITLE/',TITLE]),...
 INQUIRE PL 'IF YOU WANT A HARDCOPY ENTER 0. OTHERWISE, ENTER 1.',...
 IF PL=0, EXECUTE('PLOT.HARDCOPY'),END,...
 ERASE,...
 CLEAR TITLE ,...
END,...
...//
END
CLEAR ANMENU Z ZZ I R C PL
BUILD,TOP,MAT // for some reason, MATRIXX wants this done.
//
// End of CGTPIKF.ANALYZE command file
//
RETURN
```

```
// LTR command file
//
//**************************************************************************
//
// This command file allows the user to pursue Loop Transmission Recovery
//  after the Kalman filter has been combined with the CGT/PI controller.
//
// This command file written by Captain Steve Payson, AFIT GE/89M.
// Version 1.0  (7 Feb)
//**************************************************************************
//
DISP('LOOP TRANSMISSION RECOVERY IS PERFORMED BY ARTIFICIALLY ADDING')
DISP('PSEUDONOISE OF STRENGTH (QLTR **2) * V TO THE SYSTEM AT THE')
DISP('POINT OF ENTRY OF THE CONTROL.')
DISP('THE EQUATION WHICH CREATES THE NEW VALUE OF NOISE IS:')
DISP(' ')
DISP('                  QPRIME = QDM + QLTR**2 *BDM*V*BDM(TRANSPOSE)')
DISP(' ')
PAUSE
DISP('YOU MAY NOW SPECIFY THE V MATRIX AND THE SCALAR QUANTITY QLTR.'),...
DISP('V IS COMMONLY THE IDENTITY MATRIX, BUT THIS IS NOT REQUIRED.'),...
DUMMY=0;
WHILE DUMMY<1,...
  DISP('V MUST BE SQUARE WITH THE DIMENSIONS OF'),NINPUTS,...
  INQUIRE V 'ENTER V: ',...
  [I,J]=SIZE(V);...
  IF I<>NINPUTS, DISP('WRONG NUMBER OF ROWS.  TRY AGAIN'),...
   ELSE IF J=NINPUTS, DUMMY=1;...
```

```
    ELSE DISP('WRONG NUMBER OF COLUMNS.  TRY AGAIN.'),...
  END,END,...
END
//
DUMMY=0;
WHILE DUMMY<1,...
  DISP('QLTR MUST BE A SCALAR QUANTITY'),...
  INQUIRE QLTR 'ENTER QLTR: ',...
  [I,J]=SIZE(QLTR);...
  IF I<>1, DISP('WRONG NUMBER OF ROWS.  TRY AGAIN'),...
   ELSE IF J=1, DUMMY=1;...
    ELSE DISP('WRONG NUMBER OF COLUMNS.  TRY AGAIN.'),...
  END,END,...
END
//
// Now generate the new value of Q, which is called QPRIME.  Also
//  find the Kalman filter gains.
//
QPRIME= QDM + (QLTR**2) *BDM*V*BDM';
[SDMD]=DISCRETIZE(SDM,NSDM,DT);...//Generate ADMD
[ADMD,BDMD,CDMD,DDMD]=SPLIT(SDMD,NSDM);...
QD1=ADMD*GDM*QPRIME*GDM'*ADMD';...// Note that QDMD incorporates the G
term
QPRIMED=(QD1+GDM*QPRIME*GDM')*.5*DT;...//From Maybeck Vol I, eq 6-118
[KFEVAL1,KFGAIN,PFMINUS1]=DESTIMATOR(ADMD,HDM,QPRIMED,RDM,QRDM);...
//
//KFGAIN as generated by DESTIMATOR is a composite of measurement
// and update equation. This next step changes it to just measurement.
```

```
//
KFGAIN=INV(ADMD)*KFGAIN;
DISP('THE VALUE OF QPRIME, AND THE NEW VALUE OF KFGAIN, ARE')
QPRIME,PAUSE,KFGAIN,PAUSE
//
CLEAR QD1 I J KFEVAL1 PFMINUS1 QPRIMED
// End of LTR command file
//
RETURN
```

```
// ANALYZE.YTMF command file
//
//*************************************************************************
//
// The only reason this command file exists is because MATRIXX doesn't
//  like mixing SYSTEM BUILD commands in with regular MATRIXX commands.
//
// This command file is called by the CGTPIKF.ANALYZE command file.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (6 Jan)
//*************************************************************************
//
BUILD
Analyze Super-Block
YTMF
//
// End of ANALYZE.YTMF command file
//
RETURN



// ANALYZE.XTMF command file
//
//*************************************************************************
//
```

```
// The only reason this command file exists is because MATRIXX doesn't
//  like mixing SYSTEM BUILD commands in with regular MATRIXX commands.
//  When I tried putting these three lines of code in the CGTPI.ANALYZE
//  command file, MATRIXX threw-up all over me. Here, XTMF is the super-
//  block which represents the truth model driven by the control law
//  consisting of a CGT/PI controller with Kalman filter in the loop.
//  These states are NOT the states of the uncontrolled truth model.
//
// This command file is called by the CGTPIKF.ANALYZE command file.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (28 Jan)
//*************************************************************************
//
BUILD
Analyze Super-Block
XTMF
//
// End of ANALYZE.XTMF command file
//
RETURN




// ANALYZE.ACTF command file
//
//*************************************************************************
```

```
//
// The only reason this command file exists is because MATRIXX doesn't
//  like mixing SYSTEM BUILD commands in with regular MATRIXX commands.
//  When I tried putting these three lines of code in the CGTPI.ANALYZE
//  command file, MATRIXX threw-up all over me.
//
// This command file is called by the CGTPI.ANALYZE command file.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (11 Jan)
//*************************************************************************
//
BUILD
Analyze Super-Block
ACTF
//
// End of ANALYZE.ACTF command file
//
RETURN
```

```
// CONTROL.WITHKF command file
//
//**********************************************************************
//
// This command file accesses MATRIXX SYSTEM BUILD and puts together
//  a series of super-blocks which generate the incremental form
//  CGT/PI with Kalman filter control law.  It calls several
//  command files to build the required super-blocks.
//
// Command files called:
//   GAIN1F.BLOCKS, GAIN1F.SUM, GAIN1F.CONNECT2, GAIN1F.CONNECT1
//   BLOCK1.BLOCKS, BLOCK1.SUM, BLOCK1.CONNECT2, BLOCK1.CONNECT1
//   BLOCK5.BLOCKS, BLOCK5.SUM, BLOCK5.CONNECT2, BLOCK5.CONNECT1
//   BLOCK6.BLOCKS, BLOCK6.SUM, BLOCK6.CONNECT2, BLOCK6.CONNECT1
//   LAWF.BLOCKS, LAWF.SUM, LAWF.CONNECT2, LAWF.CONNECT1
//   YTM.2, YTM.1, XTMD.2, XTMD.1, YCMD.1, YCMD.2
//
//
// Written by Captain Steve Payson, 1988.
// Version 1.0  (6 Feb)
//**********************************************************************
//
// Prompt the user for the command model initial conditions
//  (Truth and design model IC's are requested in KF.BUILD)
//
J=0;
WHILE J<>NSCM, DISP('INPUT THE COMMAND MODEL STATE VECTOR INITIAL'),...
  DISP('CONDITIONS. YOU MUST HAVE 1 COLUMN AND THE SAME NUMBER OF ROWS'),...
  DISP('AS THE NUMBER OF STATES'),NSCM,...
  INQUIRE XCMO 'XCMO = ',...
  [J,I]=SIZE(XCMO);...
  IF I<>1, J=0,END,... // J is the flag, so if the user didn't input 1
...                    //  column, J is set to zero and this routine repeats
END
//
// Define some terms to be used in the ACT super-block, which is built
//  regardless of whether actuator dynamics are present or not.
//  SACTHOT is a State space matrix representing the ACTuator Higher
//  Order Terms, and is square with dimensions of NSACT.
//  SACT1 is a state space matrix representing the
//  actuator first order terms, and is square with dimensions of 2*NINPUTS.
//  ACT was generated in the CONTROL command file, but this allows it
//  to be written over.
//
WINDUPFLAG=0;
IF NSACT=0, ... // That is, no actuator dynamics
  SACT1=EYE(NINPUTS);SACTHOT=SACT1;...
ELSE ... // That is, derivatives of actuators exist in the truth model
  SACT1= [ATM((NSTM-NSACT+1):(NSTM-NSACT+NINPUTS),...
              (NSTM-NSACT+1):(NSTM-NSACT+NINPUTS)),...
          BTM((NSTM-NSACT+1):(NSTM-NSACT+NINPUTS),:);...
          EYE(NINPUTS),0*EYE(NINPUTS)];...
  IF NSACT=NINPUTS,... // That is, only one derivative of actuators
    SACTHOT=EYE(NINPUTS);...
  ELSE
    SACTHOT= [ATM((NSTM-NSACT+NINPUTS+1):NSTM,(NSTM-NSACT+NINPUTS+1):NSTM),..
```

```
              BTM((NSTM-NSACT+NINPUTS+1):NSTM,:);...
              EYE(NINPUTS),0*ONES(NINPUTS,NSACT-NINPUTS)];...
  END,...
END
//
// Allow the user to define position and rate limits of actuators, and
//  turn antiwindup compensation on.
//
DISP('IF YOU WISH TO IMPOSE ACTUATOR LIMITS,')
INQUIRE LIMFLAG 'ENTER A 0. OTHERWISE, ENTER 1 '
IF LIMFLAG=1, DISP('ANY FORMER VALUES OF POSITION AND RATE LIMITS ARE'),...
 DISP('NOW BEING ERASED.'),...
...//
...// If no limits desired, set them to an arbitrary value for use
...//  in ACT.BLOCKS*.  They will be removed in ACT.NOLIMITS.
...//
PLOWLIM=ONES(1,NINPUTS);PUPLIM=2*PLOWLIM;RLOWLIM=PLOWLIM;RUPLIM=PUPLIM;...
...//
ELSE,... // that is, LIMFLAG=0
 DUMMY=0;...
 WHILE DUMMY<1,...
  DISP('YOU MAY NOW ENTER ACTUATOR POSITION AND RATE LIMITS.'),...
  DISP('FIRST, ENTER THE POSITION LOWER LIMIT AS A ROW VECTOR, WITH THE'),...
  DISP('NUMBER OF COLUMNS EQUAL TO THE NUMBER OF INPUTS,'),NINPUTS,...
  INQUIRE PLOWLIM 'POSITION LOWER LIMITS (PLOWLIM)= ',...
  [I,J]=SIZE(PLOWLIM); IF I>1, DISP('WRONG NUMBER OF ROWS. TRY AGAIN.'),...
   ELSE IF J=NINPUTS, DUMMY=1;...
    ELSE DISP('WRONG NUMBER OF COLUMNS. TRY AGAIN.'),...
 END,END,...
END,...
DUMMY=0;...
WHILE DUMMY<1,...
 DISP('NOW ENTER THE POSITION UPPER LIMIT AS A ROW VECTOR, WITH THE'),...
 DISP('NUMBER OF COLUMNS EQUAL TO THE NUMBER OF INPUTS,'),NINPUTS,...
 INQUIRE PUPLIM 'POSITION UPPER LIMITS (PUPLIM)= ',...
 [I,J]=SIZE(PUPLIM); IF I>1, DISP('WRONG NUMBER OF ROWS. TRY AGAIN.'),...
  ELSE IF J=NINPUTS, DUMMY=1;...
   ELSE DISP('WRONG NUMBER OF COLUMNS. TRY AGAIN.'),...
 END,END,...
END,... // end of generating position limits
...//
...// Generate rate limits.  If no actuator dynamics exist, the limits
will
...//  the same as the position limits, which is valid because they are
...//  cascaded together.
...//
IF NSACT=0, RLOWLIM=PLOWLIM; RUPLIM=PUPLIM;...
ELSE ...
 DUMMY=0;...
 WHILE DUMMY<1,...
  DISP('NOW ENTER THE RATE LOWER LIMIT AS A ROW VECTOR, WITH THE'),...
  DISP('NUMBER OF COLUMNS EQUAL TO THE NUMBER OF INPUTS,'),NINPUTS,...
  INQUIRE RLOWLIM 'RATE LOWER LIMITS (RLOWLIM)= ',...
  [I,J]=SIZE(RLOWLIM); IF I>1, DISP('WRONG NUMBER OF ROWS. TRY AGAIN.'),...
   ELSE IF J=NINPUTS, DUMMY=1;...
    ELSE DISP('WRONG NUMBER OF COLUMNS. TRY AGAIN.'),...
   END,END,...
  END,...
  DUMMY=0;...
  WHILE DUMMY<1,...
   DISP('NOW ENTER THE RATE UPPER LIMIT AS A ROW VECTOR, WITH THE'),...
   DISP('NUMBER OF COLUMNS EQUAL TO THE NUMBER OF INPUTS,'),NINPUTS,...
   INQUIRE RUPLIM 'RATE UPPER LIMITS (RUPLIM)= ',...
   [I,J]=SIZE(RUPLIM); IF I>1, DISP('WRONG NUMBER OF ROWS. TRY AGAIN.'),...
    ELSE IF J=NINPUTS, DUMMY=1;...
     ELSE DISP('WRONG NUMBER OF COLUMNS. TRY AGAIN.'),...
   END,END,...
  END,...
 END,... // End of generating rate limits
 DISP('IF YOU WANT TO TURN ANTIWINDUP COMPENSATION ON,'),...
 INQUIRE WINDUPFLAG 'ENTER 1. OTHERWISE, ENTER 0 : ',...
END
//
// End of routine allowing actuator limits and antiwindup compensation
//
CLEAR I J
//
// TMNMEAS, TMNINPUTS, and TMNOUTPUTS are used in the following command
//  files, but at the present time they are all equal to NINPUTS.
//  Define these variables now, but clear them at the end of this
//  command file to reduce the number of names.
//
I=NINPUTS;TMNOUTPUTS=I;TMNINPUTS=I;TMNMEAS=NMEAS;
CLEAR I
```

```
//
// Generate the Kalman Filter
//
EXECUTE('KF.BUILD')
//
//*******************************************************************
// Generate the control law.  This involves building several blocks.
// The truth and design models were discretized in KF.BUILD
//
// Generate the internal blocks of all super-blocks
//
EXECUTE('KZUDM.BLOCKS')
EXECUTE('GAIN1F.BLOCKS')
EXECUTE('BLOCK5.BLOCKS')
IF WINDUPFLAG=1, EXECUTE('LAWF.BLOCKS1'),...
 ELSE EXECUTE('LAWF.BLOCKS0'),END
//
// Generate the internal and external connections of all super-blocks
//
IF NINPUTS>1, EXECUTE('KZUDM.CONNECT2'),...
  EXECUTE('GAIN1F.CONNECT2'), EXECUTE('BLOCK5.CONNECT2'),...
  EXECUTE('LAWF.CONNECT2'),...
 ELSE EXECUTE('KZUDM.CONNECT1'), ...
  EXECUTE('GAIN1F.CONNECT1'), EXECUTE('BLOCK1.CONNECT1'),...
  EXECUTE('LAWF.CONNECT1'),...
END
FOR N=1:NSDM, EXECUTE('LAWF.INTCON'),EXECUTE('BLOCK5.INTCON'),END
//
```

```
// Generate the summing junction for all super-blocks (BLOCK6 doesn't
//  have a summing junction)
//
FOR N=1:NSDM, EXECUTE('GAIN1F.SUM'), END
FOR N=1:NINPUTS, EXECUTE('BLOCK5.SUM'),...
  EXECUTE('LAWF.SUM'),...
END
//
// Generate super-block ACTF.   ACTF represents actuator
//  dynamics and limits. It includes LAWF as an internal block.
//
DUMMY=NSTM-NSACT; // used in YTM.1 and YTM.2 command files
IF NSACT=0, EXECUTE('ACTF.BLOCKS0')
IF NSACT=NINPUTS, EXECUTE('ACTF.BLOCKS1')
IF NSACT>NINPUTS, EXECUTE('ACTF.BLOCKS2')
IF LIMFLAG=1, EXECUTE('ACTF.NOLIMITS')
IF NINPUTS>1,  EXECUTE('ACTF.CONNECT2'),...
 ELSE  EXECUTE('ACTF.CONNECT1'),...
END
//
// Define the state space matrices used in YTMF.* and XTMF.*.
// First, case of no actuator dynamics
IF NSACT=0,...
  SY=[ATM,BTM;CTM,DTM];SX=[ATM,BTM;EYE(NSTM),0*ONES(NSTM,NINPUTS)];
// Case of only first order actuator dynamics
IF NSACT=NINPUTS, SY=[ATM(1:DUMMY,:);CTM(1:NINPUTS,1:DUMMY),DTM];...
      SX=[ATM(1:DUMMY,:);EYE(DUMMY),0*ONES(DUMMY,NINPUTS)];
// Case of higher than first order actuator dynamics
```

```
IF NSACT>NINPUTS, SY=[ATM(1:DUMMY,1:DUMMY+NINPUTS);...
         CTM(1:NINPUTS,1:DUMMY),DTM];...
      SX=[ATM(1:DUMMY,1:DUMMY+NINPUTS);...
         EYE(DUMMY),0*ONES(DUMMY,NINPUTS)];...
END
//
// Generate super-block XTMF, the controlled system's state response,
//  with the Kalman filter cascaded with the CGT/PI controller.
//  Also generate YTMF, the controlled system's output response.
//  YTMF is the combination of the control law and the continuous
//  truth model (including ACT), that is, the controlled system's outputs.
//
IF NINPUTS>1, EXECUTE('XTMF.2'), EXECUTE('YTMF.2'),...
  ELSE EXECUTE('XTMF.1'), EXECUTE('YTMF.1'),END
//
// Turn SYSTEM BUILD's PLOT back on, because it was turned off in
//  BLOCK1.BLOCKS
//
BUILD,Commands,Set Auto PLOT ON, Top,Mat
CLEAR LIMFLAG WINDUPFLAG SX SY SACTHOT SACT1 DUMMY TMNOUTPUTS TMNMEAS
CLEAR TMNINPUTS
//
// End of CONTROL.WITHKF command file
//
RETURN
```

```
// KF.BUILD command file
//
//***************************************************************************
//
// This command file calls other command files which use MATRIXX SYSTEM
//  BUILD command to build the Kalman filter.  The files it calls are:
//  KF.BLOCKS, KF.SUM1, KF.SUM2, KF.CONNECT2, KF.CONNECT1, KF.EXTCON,
//  XHATMINUS.BLOCKS,  XHATMINUS.SUM1, and XHATMINUS.SUM2. It is called
by
//  the KF command file.  It is patterned after the CONTROL command
//  file in the CGTPI portion of this program.  Refer to that program
//  for programming explanations.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (30 Jan)
//***************************************************************************
//
J=0;
WHILE J<>NSTM, DISP('INPUT THE TRUTH MODEL STATE VECTOR INITIAL'),...
  DISP('CONDITIONS.  YOU MUST HAVE 1 COLUMN AND THE SAME NUMBER OF'),...
  DISP('ROWS AS THE NUMBER OF STATES'),NSTM,...
  INQUIRE XTMO 'XTMO = ',...
  [J,I]=SIZE(XTMO);...
  IF I<>1, J=0, END,...
END
//
J=0;
WHILE J<>NSDM, DISP('INPUT THE DESIGN MODEL STATE VECTOR INITIAL'),...
```

```
  DISP('CONDITIONS.  YOU MUST HAVE 1 COLUMN AND THE SAME NUMBER OF'),...
  DISP('ROWS AS THE NUMBER OF STATES'),NSDM,...
  INQUIRE XDMO 'XDMO = ',...
  [J,I]=SIZE(XDMO);...
  IF I<>1, J=0, END,...
END
//
CLEAR I J
DISP('THIS MAY TAKE A WHILE.  PLEASE BE PATIENT.')
//
//****************************************************************************
// Generate the Kalman filter
//
[STM]=[ATM,BTM;CTM,DTM];
[STMD]=DISCRETIZE(STM,NSTM,DT);
[ATMD,BTMD,CTM,DTM]=SPLIT(STMD,NSTM);
[SDMD]=DISCRETIZE(SDM,NSDM,DT);
[ADMD,BDMD,CDM,DDM]=SPLIT(SDMD,NSDM);
//
EXECUTE('KF.BLOCKS'), EXECUTE('XHATMINUS.BLOCKS')
//
IF NINPUTS>1, EXECUTE('KF.CONNECT2'),...
  ELSE EXECUTE('KF.CONNECT1'),...
END
//
FOR N=1:NINPUTS, EXECUTE('XHATMINUS.SUM1'), EXECUTE('KF.EXTCON'),END
FOR N=1:NMEAS, EXECUTE('KF.SUM1'),END
FOR N=1:NSDM, EXECUTE('KF.SUM2'), EXECUTE('XHATMINUS.SUM2'), END
```

```
//
BUILD, Commands, Set Auto PLOT ON, Top Mat
//
// End of KF.BUILD command file
//
RETURN
```

```
// KZUDM.BLOCKS command file
//
//**************************************************************************
//
// This command file is part 1 of 3 which generates super-block KZUDM
for
//  the closed loop CGT/PI/KF simulation.  It uses MATRIXX SYSTEM BUILD
//  commands.  This super-block is part of the LAWF super-block.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (28 Jan)
//**************************************************************************
//
 BUILD
 Commands
 Set Auto PLOT OFF // turn plotter off
 Edit Super-Block
 KZUDM // name of super-block
 DT  // sample period
 0   // first sample time
//
 Define Block
 1
 Dynamic Systems
 State-Space System
 Block Name
 DDM
 Number of Inputs
```

```
NINPUTS
Number of Outputs
NINPUTS
Number of States
0
Parameter Entry
DDM
//
Define Block
2
Dynamic Systems
Time Delay: exp(-kTs)
Block Name
DELAY
Number of Inputs
NINPUTS
Number of Outputs
NINPUTS
Parameter Entry
DT
//
Define Block
3
Dynamic Systems
State-Space System
Block Name
KZ
Number of Inputs
```

```
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 0
 Parameter Entry
 KZ
//
 TOP
 MAT
//
// End of KZUDM.BLOCKS command file
//
RETURN




// KZUDM.CONNECT2 command file
//
//***************************************************************************
//
// This command file is part 2 of 3 which generates super-block BLOCK6
for
//  the closed loop CGT/PI/KF simulation.  It uses MATRIXX SYSTEM BUILD
//  commands.  This super-block is part of the LAWF super-block.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (28 Jan)
//****************************************************************************
//
 BUILD
 Edit Super-Block
 KZUDM // name of super-block
 0  // sample period same as in BLOCK6.BLOCKS
//
 Connect Blocks
 Internal Path   // connect blocks 1 to 2
 1
 2
 Y  // the connection is simple
 Internal Path
 2
 3
 Y
 External Input
 NINPUTS // dimension of external input vector
 1 // external input goes into block 1
 Y
 External Output
 NINPUTS // dimension of external output vector
 3 // external output comes out of block 3
 Y
//
 TOP
 MAT
```

```
//
// End of KZUDM.CONNECT2 command file
//
RETURN




// KZUDM.CONNECT1 command file
//
//****************************************************************************
//
// This command file is part 3 of 3 which generates super-block BLOCK6
for
//  the closed loop CGT/PI/KF simulation.  It uses MATRIXX SYSTEM BUILD
//  commands.  This super-block is part of the LAWF super-block.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (28 Jan)
//****************************************************************************
//
 BUILD
 Edit Super-Block
 KZUDM // name of super-block
 0  // sample period same as in BLOCK6.BLOCKS
//
 Connect Blocks
 Internal Path   // connect blocks 1 to 2
 1
 2
 Internal Path
 2
 3
 External Input
 NINPUTS // dimension of external input vector
 1 // external input goes into block 1
 External Output
 NINPUTS // dimension of external output vector
 3 // external output comes out of block 3
//
 TOP
 MAT
//
// End of KZUDM.CONNECT1 command file
//
RETURN




// GAIN1F.BLOCKS command file
//
//****************************************************************************
//
// This command file is part 1 of 4 which generates super-block GAIN1F
for
```

```
//  the closed loop CGT/PI/KF simulation.  It uses MATRIXX SYSTEM BUILD
//  commands.  GAIN1F is part of super-block BLOCK5.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (28 Jan)
//******************************************************************************
//
 BUILD
 Edit Super-Block
 GAIN1F // name of super-block
 DT // sample period
 0 // first sample time
//
 Define Block
 1
 Dynamic Systems
 Time Delay: exp(-kTs)
 Block Name
 DELAY
 Number of Inputs
 NSDM
 Number of Outputs
 NSDM
 Parameter Entry
 DT
//
 Define Block
 3
 Dynamic Systems
 State-Space System
 Block Name
 KX
 Number of Inputs
 NSDM
 Number of Outputs
 NINPUTS
 Number of States
 0
 Parameter Entry
 KX
//
 Define Block
 2
 Algebraic Equations
 Sum of Vectors
 Number of Outputs
 NSDM
 Parameter Entry
 2
 -1, 1
//
 TOP
 MAT
//
// End of GAIN1F.BLOCKS command file
//
```

```
RETURN




// GAIN1F.SUM command file
//
//***************************************************************************
//
// This command file is part 2 of 4 which generates super-block GAIN1F
for
//  the closed loop CGT/PI/KF simulation.  It uses MATRIXX SYSTEM BUILD
//  commands.  GAIN1F is part of super-block BLOCK5.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (28 Jan)
//***************************************************************************
//
 BUILD
 Edit Super-Block
 GAIN1F // name of super-block
 0 // sample period same as in GAIN1F.BLOCKS
//
 Connect Blocks
 Internal Path  // block 2 is a summing junction which sums the external
//                 input and the output of the time delay, block 1
 1
 2
 N,N
 0,0
 External Input
 NSDM
 2
 N,(N+NSDM)
 0,0
//
 TOP
 MAT
//
// End of GAIN1F.SUM command file
//
RETURN




// GAIN1F.CONNECT2 command file
//
//****************************************************************************
//
// This command file is part 3 of 4 which generates super-block GAIN1F
for
//  the closed loop CGT/PI/KF simulation.  It uses MATRIXX SYSTEM BUILD
//  commands.  GAIN1F is part of super-block BLOCK5.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (28 Jan)
//*************************************************************************
//
 BUILD
 Edit Super-Block
 GAIN1F // name of super-block
 0 // sample period same as in GAIN1.BLOCKS
//
 Connect Blocks
 Internal Path
 2
 3
 Y
 External Input
 NSDM
 1
 Y
 External Output
 NINPUTS
 3
 Y
//
 TOP
 MAT
//
// End of GAIN1F.CONNECT2 command file
//
RETURN
```

```
// GAIN1F.CONNECT1 command fil•
//
//*****************************************************************************
//
// This command file is part 4 of 4 which generates super-block GAIN1F
for
//  the closed loop CGT/PI/KF simulation.  It uses MATRIXX SYSTEM BUILD
//  commands.  GAIN1F is part of super-block BLOCK5.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (28 Jan)
//*****************************************************************************
//
 BUILD
 Edit Super-Block
 GAIN1F // name of super-block
 0 // sample period same as in GAIN1F.BLOCKS
//
 Connect Blocks
 Internal Path
 2
 3
 External Input
 NSDM
 1
 External Output
 NINPUTS
 3
//
 TOP
 MAT
//
// End of GAIN1F.CONNECT1 command file
//
RETURN




// BLOCK5.BLOCKS command file
//
//***************************************************************************
//
// This command file is part 1 of 5 which generates super-block BLOCK5
for
//  the closed loop CGT/PI/KF simulation.  It uses MATRIXX SYSTEM BUILD
//  commands.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (28 Jan)
//***************************************************************************
//
```

```
 BUILD
 Edit Super-Block
 BLOCK5 // super-block name
 DT // sample period
 0  // first sample time
//
 Define Block
 1
 Super-Block
 Block Name
 GAIN1F
 Number of Inputs
 NSDM
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 2
 Super-Block
 Block Name
 KZUDM
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
```

```
 Define Block
 4
 Dynamic Systems
 Time Delay: exp(-kTs)
 Block Name
 DELAY
 Number of Inputs
 NSDM
 Number of Outputs
 NSDM
 Parameter Entry
 DT
//
 Define Block
 5
 Dynamic Systems
 State-Space System
 Block Name
 CDM
 Number of Inputs
 NSDM
 Number of Outputs
 NINPUTS
 Number of States
 0
 Parameter Entry
 CDM
//
```

```
 Define Block
 6
 Dynamic Systems
 State-Space System
 Block Name
 KZ
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 0
 Parameter Entry
 KZ
//
 Define Block
 3
 Algebraic Equations
 Sum of Vectors
 Number of Outputs
 NINPUTS
 Parameter Entry
 3 // number of input vectors
 1,1, 1 // sign on vectors
//
 TOP
 MAT
//
```

```
// End of BLOCK5.BLOCKS command file
//
RETURN




// BLOCK5.SUM command file
//
//*****************************************************************************
//
// This command file is part 2 of 5 which generates super-block BLOCK5
for
//  the closed loop CGT/PI/KF simulation.  It uses MATRIXX SYSTEM BUILD
//  commands.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (28 Jan)
//*****************************************************************************
//
 BUILD
 Edit Super-Block
 BLOCK5 // super-block name
 0 // sample period same as in BLOCK5.BLOCKS
//
 Connect Blocks
 External Input
 NSDM+NINPUTS
 2
 N,N
 0,0
 Internal Path
 1
 3
 N,N
 0,0
 Internal Path
 6
 3
 N,(N+NINPUTS)
 0,0
 Internal Path
 2
 3
 N,(N+2*NINPUTS)
 0,0
//
 TOP
 MAT
//
// End of BLOCK5.SUM command file
//
RETURN
```

```
// BLOCK5.CONNECT2 command file
//
//***************************************************************************
//
// This command file is part 3 of 5 which generates super-block BLOCK5
for
//  the closed loop CGT/PI/KF simulation.  It uses MATRIXX SYSTEM BUILD
//  commands.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (28 Jan)
//***************************************************************************
//
 BUILD
 Edit Super-Block
 BLOCK5 // super-block name
 0 // sample period same as in BLOCK5.BLOCKS
//
 Connect Blocks
 Internal Path
 5
 6
 Y
 Internal Path
 4
 5
 Y
 External Output
 NINPUTS
 3
 Y
//
 TOP
 MAT
//
// End of BLOCK5.CONNECT2 command file
//
RETURN




// BLOCK5.CONNECT1 command file
//
//****************************************************************************
//
// This command file is part 3 of 5 which generates super-block BLOCK5
for
//  the closed loop CGT/PI/KF simulation.  It uses MATRIXX SYSTEM BUILD
//  commands.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (28 Jan)
//****************************************************************************
//
```

```
 BUILD
 Edit Super-Block
 BLOCK5 // super-block name
 0 // sample period same as in BLOCK5.BLOCKS
//
 Connect Blocks
 Internal Path
 5
 6
 Internal Path
 4
 5
 External Output
 NINPUTS
 3
//
 TOP
 MAT
//
// End of BLOCK5.CONNECT1 command file
//
RETURN
```

```
// BLOCK5.INTCON command file
//
```

```
//**************************************************************************
//
// This command file is part 5 of 5 which generates super-block BLOCK5
for
//  the closed loop CGT/PI/KF simulation.  It uses MATRIXX SYSTEM BUILD
//  commands.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (28 Jan)
//**************************************************************************
//
 BUILD
 Edit Super-Block
 BLOCK5 // super-block name
 0 // sample period same as in BLOCK5.BLOCKS
//
 Connect Blocks
 External Input
 NSDM+NINPUTS
 1
 N+NINPUTS,N // from, to
 0,0
 External Input
 NSDM+NINPUTS
 4
 N+NINPUTS,N // from, to
 0,0
//
```

```
 TOP
 MAT
//
// End of BLOCK5.INTCON command file
//
RETURN
```

```
// LAWF.BLOCKSO command file
//
//****************************************************************************
//
// This command file is part 1 of 6 which generates super-block LAWF for
//  the closed loop CGT/PI/KF simulation.  It uses MATRIXX SYSTEM BUILD
//  commands.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (1 Feb)
//****************************************************************************
//
 BUILD
 Edit Super-Block
 LAWF  // super-block name
 DT // sample period
 0  // first sample period
//
 Define Block
 1
 Super-Block
 Block Name
 BLOCK1
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 2
 Dynamic Systems
 Time Delay: exp(-kTs)
 Block Name
 DELAY
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
 DT
//
 Define Block
 6
 Super-Block
 Block Name
 BLOCK5
 Number of Inputs
 NSDM+NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 5
```

```
Super-Block
Block Name
FILTER
Number of Inputs
NINPUTS
Number of Outputs
NSDM
Parameter Entry
//
Define Block
4 // block location; this block formerly represented antiwindup compensation
Dynamic Systems
State-Space System
Block Name
WINDUP
Number of Inputs
NINPUTS
Number of Outputs
NINPUTS
Number of States
0
Parameter Entry
EYE(NINPUTS)
//
Define Block
3
Algebraic Equations
Sum of Vectors
```

```
 Number of Outputs
 NINPUTS
 Parameter Entry
 3
 1,1,-1,
//
 TOP
 MAT
//
// End of LAWF.BLOCKS0 command file
//
RETURN




// LAWF.BLOCKS1 command file
//
//*****************************************************************************
//
// This command file is part 2 of 6 which generates super-block LAWF for
//  the closed loop CGT/PI/KF simulation.  It uses MATRIXX SYSTEM BUILD
//  commands.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (1 Feb)
//*****************************************************************************
//
```

```
BUILD
Edit Super-Block
LAWF  // super-block name
DT // sample period
0  // first sample period
//
Define Block
1
Super-Block
Block Name
BLOCK1
Number of Inputs
NINPUTS
Number of Outputs
NINPUTS
Parameter Entry
//
Define Block
2
Dynamic Systems
Time Delay: exp(-kTs)
Block Name
DELAY
Number of Inputs
NINPUTS
Number of Outputs
NINPUTS
Parameter Entry
```

```
DT
//
Define Block
6
Super-Block
Block Name
BLOCK5
Number of Inputs
NSDM+NINPUTS
Number of Outputs
NINPUTS
Parameter Entry
//
Define Block
5
Super-Block
Block Name
FILTER
Number of Inputs
NINPUTS
Number of Outputs
NSDM
Parameter Entry
//
Define Block
4 // block location. This block will be the antiwindup compensation
Piece-Wise Linear
L - U Bounded Limit
```

```
 Block Name
 WINDUP
 Number of Outputs
 NINPUTS
 Parameter Entry
 PLOWLIM
 PUPLIM
//
 Define Block
 3
 Algebraic Equations
 Sum of Vectors
 Number of Outputs
 NINPUTS
 Parameter Entry
 3
 1,1,-1,
//
 TOP
 MAT
//
// End of LAWF.BLOCKS1 command file
//
RETURN
```

```
// LAWF.SUM command file
//
//***************************************************************************
//
// This command file is part 3 of 6 which generates super-block LAWF for
//  the closed loop CGT/PI/KF simulation.  It uses MATRIXX SYSTEM BUILD
//  commands.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (1 Feb)
//***************************************************************************
//
 BUILD
 Edit Super-Block
 LAWF  // super-block name
 0 // sample period same as in LAWF.BLOCKS
//
 Connect Blocks
 Internal Path
 1
 3
 N,N
 0,0
 Internal Path   // Not a simple connection
 4
 6
 N,N
 0,0
 Internal Path
 2
 3
 N,(N+NINPUTS)
 0,0
 Internal Path
 6
 3
 N,(N+2*NINPUTS)
 0,0
//
 TOP
 MAT
//
// End of LAWF.SUM command file
//
RETURN




// LAWF.CONNECT2 command file
//
//****************************************************************************
//
// This command file is part 4 of 6 which generates super-block LAWF for
//  the closed loop CGT/PI/KF simulation.  It uses MATRIXX SYSTEM BUILD
//  commands.
```

```
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (1 Feb)
//**********************************************************************
//
 BUILD
 Edit Super-Block
 LAWF  // super-block name
 0 // sample period same as in LAWF.BLOCKS
//
 Connect Blocks
 External Input
 NINPUTS
 1
 Y
 Internal Path
 3
 4
 Y
 Internal Path
 4
 5
 Y
 Internal Path
 4
 2
 Y
 External Output
 NINPUTS
 4
 Y
//
 TOP
 MAT
//
// End of LAWF.CONNECT2 command file
//
RETURN




// LAWF.CONNECT1 command file
//
//***************************************************************************
//
// This command file is part 5 of 6 which generates super-block LAWF for
//  the closed loop CGT/PI/KF simulation.  It uses MATRIXX SYSTEM BUILD
//  commands.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (1 Feb)
//***************************************************************************
//
 BUILD
 Edit Super-Block
 LAWF  // super-block name
 0 // sample period same as in LAWF.BLOCKS
//
 Connect Blocks
 External Input
 NINPUTS
 1
 Internal Path
 3
 4
 Internal Path
 4
 5
 Internal Path
 4
 6
 N,N
 0,0
 External Output
 NINPUTS
 4
 Y
//
 TOP
 MAT
//
// End of LAWF.CONNECT2 command file
//
```

```
RETURN





// LAWF.INTCON command file
//
//*************************************************************************
//
// This command file is part 6 of 6 which generates super-block LAWF for
//  the closed loop CGT/PI/KF simulation.  It uses MATRIXX SYSTEM BUILD
//  commands.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (1 Feb)
//*************************************************************************
//
 BUILD
 Edit Super-Block
 LAWF  // super-block name
 0 // sample period same as in LAWF.BLOCKS
//
 Connect Blocks
 Internal Path
 5
 6
 N,N+NINPUTS
 0,0
```

```
//
 TOP
 MAT
//
// End of LAWF.INTCON command file
//
RETURN
```

```
// ACTF.BLOCKSO command file
//
//***************************************************************************
//
// This command file is 1 of 4 which allows for actuator dynamics and
//  actuator limits to be included in the simulation.  It consists of
//  MATRIXX SYSTEM BUILD commands.
//
// The name of this command file is a mnemonic representing the fact
//  that this command file builds the blocks of super-block ACTF if
//  the number of actuator states (NSACT) per actuator equals 0.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (1 Feb)
//***************************************************************************
//
 BUILD
 Edit Super-Block
 ACTF // name of super-block
 0  // sample period (continuous system)
//
 Define Block
 1 // block location; this block is the LAW super-block
 Super-Block
 Block Name
 LAWF
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 2 // block location; this block represents the actuator higher order
//     terms, that is, the terms above one derivative
 Dynamic Systems
 State-Space System
 Block Name
 ACTHOT
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 0 // If NSACT<2*ninputs, no higher order dynamics
 Parameter Entry
 SACTHOT
//
 Define Block
 3 // block location; this block represents rate limits
 Piece-Wise Linear
 L - U Bounded Limit
 Block Name
 RLIMIT
 Number of Outputs
 NINPUTS
 Parameter Entry
 RLOWLIM
 RUPLIM
//
 Define Block
 4 // block location; this block represents actuator first order terms
 Dynamic Systems
 State-Space System
 Block Name
 ACT1
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 0 // If NSACT=0, no actuator dynamics
 Parameter Entry
 SACT1
//
 Define Block
 5 // block location; this block represents position limits
 Piece-Wise Linear
 L - U Bounded Limit
 Block Name
 PLIMIT
 Number of Outputs
 NINPUTS
 Parameter Entry
 PLOWLIM
 PUPLIM
//
 TOP
 MAT
//
// End of ACTF.BLOCKS0 command file
//
RETURN




// ACTF.BLOCKS1 command file
//
//****************************************************************************
//
// This command file is 2 of 4 which allows for actuator dynamics and
//  actuator limits to be included in the simulation.  It consists of
//  MATRIXX SYSTEM BUILD commands.
//
// The name of this command file is a mnemonic representing the fact
//  that this command file builds the blocks of super-block ACTF if
//  the number of actuator states (NSACT) per actuator equals 1.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (1 Feb)
//****************************************************************************
```

```
//
 BUILD
 Edit Super-Block
 ACTF // name of super-block
 0  // sample period (continuous system)
//
 Define Block
 1 // block location
 SUPER-BLOCK
 Block Name
 LAWF
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 2 // block location; this block represents the actuator higher order
//     terms, that is, the terms above one derivative
 Dynamic Systems
 State-Space System
 Block Name
 ACTHOT
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 0 // If NSACT<2*ninputs, no higher order dynamics
 Parameter Entry
 SACTHOT
//
 Define Block
 3 // block location; this block represents rate limits
 Piece-Wise Linear
 L - U Bounded Limit
 Block Name
 RLIMIT
 Number of Outputs
 NINPUTS
 Parameter Entry
 RLOWLIM
 RUPLIM
//
 Define Block
 4 // block location; this block represents actuator first order terms
 Dynamic Systems
 State-Space System
 Block Name
 ACT1
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 NINPUTS
 Parameter Entry
 SACT1
 N // initial states not zero
 XTMO((NSTM-NSACT+1):(NSTM-NSACT+NINPUTS))
// actuator higher order term state initial conditions
//
 Define Block
 5 // block location; this block represents position limits
 Piece-Wise Linear
 L - U Bounded Limit
 Block Name
 PLIMIT
 Number of Outputs
 NINPUTS
 Parameter Entry
 PLOWLIM
 PUPLIM
//
 TOP
 MAT
//
// End of ACTF.BLOCKS1 command file
//
RETURN
```

```
// ACTF.BLOCKS2 command file
//
//*****************************************************************************
//
// This command file is 3 of 4 which allows for actuator dynamics and
//  actuator limits to be included in the simulation.  It consists of
//  MATRIXX SYSTEM BUILD commands.
//
// The name of this command file is a mnemonic representing the fact
//  that this command file builds the blocks of super-block ACTF if
//  the number of actuator states (NSACT) per actuator is =>2.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (1 Feb)
//*****************************************************************************
//
 BUILD
 Edit Super-Block
 ACTF // name of super-block
 0  // sample period (continuous system)
//
 Define Block
 2 // block location
 Super-Block
 Block Name
 LAWF
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 2 // block location; this block represents the actuator higher order
//     terms, that is, the terms above one derivative
 Dynamic Systems
 State-Space System
 Block Name
 ACTHOT
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 (NSACT-NINPUTS)
 Parameter Entry
 SACTHOT
 N // initial states not zero
 XTMO((NSTM-NSACT+NINPUTS+1):NSTM,:) // actuator higher order term state
//                                            initial conditions
//
 Define Block
 3 // block location; this block represents rate limits
 Piece-Wise Linear
 L - U Bounded Limit
 Block Name
 RLIMIT
 Number of Outputs
 NINPUTS
 Parameter Entry
 RLOWLIM
 RUPLIM
//
 Define Block
 4 // block location; this block represents actuator first order terms
 Dynamic Systems
 State-Space System
 Block Name
 ACT1
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 NINPUTS
 Parameter Entry
 SACT1
 N // initial states not zero
 XTMO((NSTM-NSACT+1):(NSTM-NSACT+NINPUTS))
// actuator higher order term state initial conditions
//
 Define Block
 5 // block location; this block represents position limits
 Piece-Wise Linear
 L - U Bounded Limit
 Block Name
 PLIMIT
 Number of Outputs
 NINPUTS
 Parameter Entry
 PLOWLIM
 PUPLIM
//
 TOP
 MAT
//
// End of ACTF.BLOCKS2 command file
//
RETURN




// ACTF.NOLIMITS command file
//
//*****************************************************************************
//
// This command file is 4 of 4 which allows for actuator dynamics and
//  actuator limits to be included in the simulation.  It consists of
//  MATRIXX SYSTEM BUILD commands.
//
```

```
// The name of this command file is a mnemonic representing the fact
//  that this command file deletes the blocks of super-block ACTF representin
//  the limits if the user chose not to use limits (LIMFLAG=0).
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (1 Feb)
//*****************************************************************************
//
 BUILD
 Edit Super-Block
 ACTF // name of super-block
 0 // the sample period is same as defined in ACT.BLOCKS*
//
 Define Block
 3 // block location; this block formerly represented rate limits
 Dynamic Systems
 State-Space System
 Block Name
 RLIMIT
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 0
 Parameter Entry
 EYE(NINPUTS)
//
```

```
 Define Block
 5 // block location; this block formerly represented position limits
 Dynamic Systems
 State-Space System
 Block Name
 PLIMIT
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Number of States
 0
 Parameter Entry
 EYE(NINPUTS)
//
 TOP
 MAT
//
// End of ACTF.NOLIMITS command file
//
RETURN




// YTMF.2 command file
//***************************************************************************
//
```

```
// This command file generates the discrete output vector of the truth
//  model if the dimension of the connections is 2 or greater.  The flag
is
//  NINPUTS, because that is the dimension of the connections.  It
//  uses MATRIXX SYSTEM BUILD commands.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (6 Feb)
//*************************************************************************
//
 BUILD
 Edit Super-Block
 YTMF // name of super-block
 0 // sample period; continuous time system
//
 Define Block
 2 // block location
 Super-Block
 Block Name
 ACTF // this super-block generates the actuator dynamics and limits
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 3 // block location. This block is the truth model
 Dynamic Systems
 State-Space System
 Block Name
 STM
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS // number of truth model outputs
 Number of States
 DUMMY
 Parameter Entry
 SY
 N // initial states not zero
 XTMO(1:DUMMY) // design model initial conditions
//
 Connect Blocks
 Internal Path
 2
 3
 Y // simple connection
 External Input
 NINPUTS // number of inputs
 2 // block location
 Y // simple connection
 External Output
 NINPUTS // number of outputs
 3 // block location
 Y // simple connection
 Describe Blocks
 TOP
 MAT
//
// End of YTMF.2 command file
//
RETURN



// YTMF.1 command file
//************************************************************************
//
// This command file generates the discrete output vector of the truth
//  model if the dimension of the connections is 2 or greater.  The flag
is
//  NINPUTS, because that is the dimension of the connections.  It
//  uses MATRIXX SYSTEM BUILD commands.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (6 Feb)
//************************************************************************
//
 BUILD
 Edit Super-Block
 YTMF // name of super-block
 0 // sample period; continuous time system
```

```
//
 Define Block
 2 // block location
 Super-Block
 Block Name
 ACTF // this super-block generates the actuator dynamics and limits
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 3 // block location. This block is the truth model
 Dynamic Systems
 State-Space System
 Block Name
 STM
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS // number of truth model outputs
 Number of States
 DUMMY
 Parameter Entry
 SY
 N // initial states not zero
 XTMO(1:DUMMY) // design model initial conditions
```

```
//
 Connect Blocks
 Internal Path
 2
 3
 External Input
 NINPUTS // number of inputs
 2 // block location
 External Output
 NINPUTS // number of outputs
 3 // block location
 Describe Blocks
 TOP
 MAT
//
// End of YTMF.1 command file
//
RETURN




// XTMF.2 command file
//***********************************************************************
//
// This command file generates the controlled discrete states of the truth
//  model if the dimension of the connections is 2 or greater.  The flag
is
```

```
//  NINPUTS, because that is the dimension of the connections.  It
//  uses MATRIXX SYSTEM BUILD commands.
//
// This command file is called by the CONTROL command file.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (6 Feb)
//************************************************************************
//
 BUILD
 Edit Super-Block
 XTMF // name of super-block
 0 // sample period
//
 Define Block
 2 // block location
 Super-Block
 Block Name
 ACTF // this super-block generates the actuator dynamics and limits
 Number of Inputs
 NINPUTS
 Number of Outputs
 NINPUTS
 Parameter Entry
//
 Define Block
 3 // block location. This block is the truth model
 Dynamic Systems
 State-Space System
 Block Name
 TMCONT
 Number of Inputs
 NINPUTS
 Number of Outputs
 DUMMY // number of truth model states less actuator states
 Number of States
 DUMMY
 Parameter Entry
 SX
 N // initial states not zero
 XTMO(1:DUMMY) // truth model initial conditions
//
 Connect Blocks
 Internal Path
 2
 3
 Y // simple connection
 External Input
 NINPUTS // number of inputs
 2 // block location
 Y // simple connection
 External Output
 DUMMY // number of outputs
 3 // block location
 Y // simple connection
 Describe Blocks
 TOP
 MAT
//
// End of XTMF.2 command file
//
RETURN




// XTMF.1 command file
//**************************************************************************
//
// This command file generates the controlled discrete states of the truth
//  model if the dimension of the connections is 2 or greater.  The flag
is
//  NINPUTS, because that is the dimension of the connections.  It
//  uses MATRIXX SYSTEM BUILD commands.
//
// This command file is called by the CONTROL command file.
//
// This command file was written by Captain Steve Payson, 1988.
// Version 1.0  (6 Feb)
//**************************************************************************
//
 BUILD
 Edit Super-Block
 XTMF // name of super-block
```

```
0 // sample period
//
Define Block
2 // block location
Super-Block
Block Name
ACTF // this super-block generates the actuator dynamics and limits
Number of Inputs
NINPUTS
Number of Outputs
NINPUTS
Parameter Entry
//
Define Block
3 // block location. This block is the truth model
Dynamic Systems
State-Space System
Block Name
TMCONT
Number of Inputs
NINPUTS
Number of Outputs
DUMMY // number of truth model states less actuator states
Number of States
DUMMY
Parameter Entry
XS
N // initial states not zero
```

```
 XTMO(1:DUMMY) // truth model initial conditions
//
 Connect Blocks
 Internal Path
 2
 3
 External Input
 NINPUTS // number of inputs
 2 // block location
 External Output
 DUMMY // number of outputs
 3 // block location
 Y // connection is simple; case of 1 truth model state not covered
 Describe Blocks
 TOP
 MAT
//
// End of XTMF.1 command file
//
RETURN
```

```
// XHATMINUS.BLOCKS command file
//
//****************************************************************************
//
// This command file is part 1 of 3 which generates the estimated x value
//  at time t sub i-1.  It is written using MATRIXX commands.  These
//  command files take the input to the FILTER super-block, which is the
//  optimal control u, feed it through the BDMD matrix (block 1), and
//  take the output of block 5 of the FILTER super-block and feed it
//  through ADMD.  These two results are added together to form xhat at
//  time t sub i-1.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (30 Jan)
//****************************************************************************
//
 BUILD
 Edit Super-Block
 XHAT- // name of super-block
 DT  // sample period
 0  // first sample time
//
 Define Block
 1
 Dynamic Systems
 State-Space System
 Block Name
 BDMD
 Number of Inputs
 NINPUTS
 Number of Outputs
 NSDM
 Number of States
 0
 Parameter Entry
 BDMD
//
 Define Block
 2
 Dynamic Systems
 State-Space System
 Block Name
 ADMD
 Number of Inputs
 NSDM
 Number of Outputs
 NSDM
 Number of States
 0
 Parameter Entry
 ADMD
//
 Define Block  // the summing junction
 3
 Algebraic Equations
 Sum of Vectors
 Number of Inputs
 NSDM
 Number of Outputs
 NSDM
 Parameter Entry
 2  // number of input vectors
 1,1 // signs on input vectors
//
 Define Block
 4 // Block location
 Dynamic Systems
 Time Delay: exp(-kTs)
 Block Name
 DELAY
 Number of Inputs
 NSDM
 Number of Outputs
 NSDM
 Parameter Entry
 DT  // amount of time delay
//
 Connect Blocks
 Internal Path
 3 // from summing junction
 4 // to time delay
 Y // connection is simple
//
 External Output
 NSDM
 4 // External output comes from block 4, the time delay
 Y // Connection is simple
//
 TOP
 MAT
//
// End of XHATMINUS.BLOCKS command file
//
RETURN




// XHATMINUS.SUM1 command file
//
//*****************************************************************************
//
// This command file is part 2 of 3 which generates the estimated x value
//  at time t sub i-1.  It is written using MATRIXX commands.  These
//  command files take the input to the FILTER super-block, which is the
//  optimal control u, feed it through the BDMD matrix (block 1), and
//  take the output of block 5 of the FILTER super-block and feed it
//  through ADMD.  These two results are added together to form xhat at
//  time t sub i-1.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (30 Jan)
```

```
//*****************************************************************************
//
 BUILD
 Edit Super-Block
 XHAT- // name of super-block
 0  // sample period same as in XHATMINUS.BLOCKS
//
 Connect Blocks
 External Input
 (NINPUTS+NSDM) // dimension of input vector
 1  // external input comes into block 1
 N,N // a connection
 0,0 // end connection mode
 TOP
 MAT
//
// End of XHATMINUS.SUM1 command file
//
RETURN




// XHATMINUS.SUM3 command file
//
//*****************************************************************************
//
// This command file is part 3 of 3 which generates the estimated x value
//  at time t sub i-1.  It is written using MATRIXX commands.  These
//  command files take the input to the FILTER super-block, which is the
//  optimal control u, feed it through the BDMD matrix (block 1), and
//  take the output of block 5 of the FILTER super-block and feed it
//  through ADMD.  These two results are added together to form xhat at
//  time t sub i-1.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (30 Jan)
//***************************************************************************
//
 BUILD
 Edit Super-Block
 XHAT- // name of super-block
 0  // sample period same as in XHATMINUS.BLOCKS
//
 Connect Blocks
 External Input
 (NINPUTS+NSDM)
 2
 (N+NINPUTS),N
 0,0
 Internal Path
 1
 3
 N,N
 0,0
 Internal Path
 2
 3
 N,(N+NSDM)
 0,0
 TOP
 MAT
//
// End of XHATMINUS.SUM2 command file
//
RETURN
```

```
// KF.BLOCKS command file
//
//***************************************************************************
//
// This command file is part 1 of 6 which builds the Kalman filter.  It
//  uses MATRIXX SYSTEM BUILD commands to do this.  The other command
//  files used to build the Kalman filter are KF.SUM, KF.CONNECT2, and
//  KF.CONNECT1.
//
// This command file is called by the KF.BUILD command file.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M.
// Version 1.0  (20 Jan)
//***************************************************************************
//
 BUILD
 Command // change menu
 Set Auto PLOT OFF // turn graphics off
 Top // change menu
 Edit Super-Block
 FILTER // name of super-block
 DT // sample period
 0 // first sample time
//
 Define Block
 1 // block location
 Dynamic Systems
 State-Space System
 Block Name
 TM.HTM // this block generates the measurement vector z
 Number of Inputs
 NINPUTS
 Number of Outputs
 TMNMEAS
 Number of States
 NSTM
 Parameter Entry
 [ATMD,BTMD;HTM,0*ONES(TMNMEAS,TMNINPUTS)]//tmnmeas better equal nmeas!!!
 N // initial states not zero
 XTMO // truth model initial conditions
//
 Define Block
 2 // block location
 Dynamic Systems
 State-Space System
 Block Name
 HDM // this block is the design model measurement matrix, H. It's output
       // is H * x(t_i-) (estimated x)
 Number of Inputs
 NSDM
 Number of Outputs
 NMEAS // IT APPEARS THAT NMEAS WILL HAVE TO EQUAL TMNMEAS
 Number of States
 0
 Parameter Entry
 HDM
```

```
//
 Define Block
 3 // block location
 Algebraic Equations
 Sum of Vectors
 Number of Outputs
 TMNMEAS
 Parameter Entry
 2 // two input vectors
 1,-1 // the sign of each respective input vector
//
 Define Block
 4 // block location
 Dynamic Systems
 State-Space System
 Block Name
 KFGAIN // this is the gain matrix of the Kalman filter
 Number of Inputs
 TMNMEAS
 Number of Outputs
 NSDM
 Number of States
 0
 Parameter Entry
 KFGAIN
//
 Define Block
 5 // block location
 Algebraic Equations
 Sum of Vectors
 Number of Outputs
 NSDM
 Parameter Entry
 2 // two input vectors
 1,1 // the sign of each respective input vector
//
 Define Block
 6 // block location
 Super-Block
 Block Name
 XHAT- // outputs of this super-block are x(t_i-) (estimated x)
 Number of Inputs
 (NSDM+NINPUTS)
 Number of Outputs
 NSDM
 Parameter Entry
//
 TOP
 MAT
//
// End of KF.BLOCKS command file
//
RETURN
```

```
// KF.CONNECT2 command file
//
//***************************************************************************
//
// This command file is part 4 of 6 which builds the Kalman filter.  The
//  other command files which help build the filter are KF.BLOCKS, KF.SUM,
//  and KF.CONNECT1.  This command file deals with the connections when
//  the number of measurements and inputs is greater than 1.  It assumes
the
//  number of states of the truth and design models are greater than one.
//  KF.CONNECT1 handles the case where the measurements and inputs are
//  equal to one.
//
// This command file is called by the KF.BUILD command file.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// VERSION 1.0  (30 Jan)
//***************************************************************************
//
 BUILD
 Edit Super-Block
 FILTER // name of super-block
 0 // same sample period as KF.BLOCKS used when generating this super-block
//
 Connect Blocks
 External Input
 NINPUTS // dimension of external input, which is the number of inputs
 1 // external input goes to block 1
 Y // connection is simple
//
 External Output
 NSDM // dimension of external output, which is the number of states
 5 // external output comes from block 5
 Y // connection is simple
//
 Internal Path        // from block 3 to 4
 3
 4
 Y // connection is simple
//
// Internal Path        // from block 5 to 6
// 5
// 6
// Y // connection is simple
//
 Internal Path        // from block 6 to 2
 6
 2
 Y // connection is simple
//
 TOP
 MAT
//
// end of KF.CONNECT2 command file
//
```

```
RETURN




// KF.CONNECT1 command file
//
//**************************************************************************
//
// This command file is part 5 of 6 which builds the Kalman filter.  The
//  other command files which help build the filter are KF.BLOCKS, KF.SUM,
//  and KF.CONNECT2.  This command file deals with the connections when
//  the number of inputs and measurements are equal to 1.  The number
//  of states of the truth and design models are assumed to be more than
//  one.  If this is not the case, this command file will not work.
//
// This command file is called by the KF.BUILD command file.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// VERSION 1.0  (20 Jan)
//**************************************************************************
//
 BUILD
 Edit Super-Block
 FILTER // name of super-block
 0 // same sample period as KF.BLOCKS used when generating this super-block
//
 Connect Blocks
 External Input
 NINPUTS // dimension of external input, which is the number of inputs
 1 // external input goes to block 1
//
 External Output
 NSDM // dimension of external output, which is the number of states
 5 // external output comes from block 5
//
 Internal Path        // from block 3 to 4
 3
 4
//
 Internal Path        // from block 5 to 6
 5
 6
 Y // connection is simple (number of states are assumed greater than
1)
//
 Internal Path        // from block 6 to 2
 6
 2
 Y // connection is simple
//
 TOP
 MAT
//
// end of KF.CONNECT1 command file
//
```

```
RETURN




// KF.EXTCON command file
//
//***************************************************************************
//
// This command file is part 6 of 6 which, using MATRIXX SYSTEM BUILD
//  commands, builds the Kalman filter.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (30 Jan)
//***************************************************************************
//
 BUILD
 Edit Super-Block
 FILTER // super-block name
 0 // same sampling time as used in KF.BLOCKS command file
//
 Connect Blocks
 External Input
 NINPUTS
 6
 N,N
 0,0
//
```

```
 TOP
 MAT
//
// End of KF.EXTCON command file
//
RETURN




// KF.SUM1 command file
//
//**************************************************************************
//
// This command file is part 2 of 6 which uses MATRIXX SYSTEM BUILD commands
//  to build the Kalman filter.  The other command files are KF.BLOCKS,
//  KF.CONNECT2, and KF.CONNECT1.  This particular command file deals
with
//  summing junctions.
//
// This command file is called by the KF.BUILD command file.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (20 Jan)
//**************************************************************************
//
 BUILD
 Edit Super-Block
 FILTER // name of super block
 0 // same sample period as KF.BLOCKS used generating this super-block
//
 Connect Blocks
 Internal Path
 1
 3
 N,N // a connection
 0,0 // end connection mode
 Internal Path
 2
 3
 N,N+NMEAS // a connection
 0,0 // end connection mode
//
 TOP
 MAT
//
// End of KF.SUM1 command file
//
RETURN




// KF.SUM2 command file
//
//***************************************************************************
```

```
//
// This command file is part 3 of 6 which uses MATRIXX SYSTEM BUILD commands
//  to build the Kalman filter.  The other command files are KF.BLOCKS,
//  KF.CONNECT2, and KF.CONNECT1.  This particular command file deals
with
//  summing junctions.
//
// This command file is called by the KF.BUILD command file.
//
// This command file was written by Captain Steve Payson, AFIT GE/89M
// Version 1.0  (20 Jan)
//***************************************************************************
//
 BUILD
 Edit Super-Block
 FILTER // name of super block
 0 // same sample period as KF.BLOCKS used generating this super-block
//
 Connect Blocks
 Internal Path
 4
 5
 N,N // a connection
 0,0 // end connection mode
//
 Internal Path
 6
 5
 N,N+NSDM // a connection
 0,0 // end connection mode
//
 Internal Path
 5
 6
 N,(N+NINPUTS)
 0,0
//
 TOP
 MAT
//
// End of KF.SUM2 command file
//
RETURN
```

Bibliography

1. ASD/ENESS. Military Specification - Flying Qualities of Piloted Aircraft. November 1980. Mil-F-8785C, Wright Patterson AFB, OH.

2. A. Y. Barraud. A Numerical Algorithm to Solve $A^TXA - X = Q$. *IEEE Trans. Automat. Control AC-22 (5)*, 883–884, Oct 1977.

3. J. Blakelock. *Automatic Control of Aircraft and Missiles.* John Wiley and Sons, Inc., New York, 1965.

4. J. J. D'Azzo and C. H. Houpis. *Linear Control Systems Analysis and Design.* McGraw-Hill Book Company, New York, 1981.

5. R. M. Floyd. *Design of Advanced Digital Flight Control Systems Via Command Generator Tracker (CGT) Synthesis Methods.* Master's thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1981.

6. E. G. Gilbert. Conditions for Minimizing the Norm Sensitivity of Characteristic roots. *Conference on Information Sciences and Systems, Baltimore, Maryland.*, March 1983.

7. G. L. Gross. *LQG/LTR Design of a Robust Flight Controller for the Stol F-15.* Master's thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1985.

8. D. Hammond. *Multivariable Control Law Design For The Control Reconfigurable Combat Aircraft (CRCA).* Master's thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1988.

9. R.A. Houston. *An LQG Up-And-Away Flight Control Design For The STOL F-15 Aircraft.* Master's thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1985.

10. Integrated Systems, Inc. *MATRIXx Reference Manual.* February 1988.

11. P. S. Maybeck. *Stochastic Models, Estimation and Control.* Volume 3, Academic Press, New York, 1982.

12. P. S. Maybeck. *Stochastic Models, Estimation and Control.* Volume 1, Academic Press, New York, 1979.

13. P. S. Maybeck, W. G. Miller, and J. M. Howey. Robustness Enhancement for LQG Digital Flight Controller design. *IEEE 1984 National Aerospace and Electronics Conference (NAECON 1984), Dayton, Ohio*, 518–525, May 1984.

14. W. G. Miller. *Robust Multivariable Controller Design Via Implicit Model-Following Methods.* Master's thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1983.

15. K.N. Neumann. *A Digital Rate Controller For The Control Reconfigurable Combat Aircraft Designed Using Quantitative Feedback Theory.* Master's thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1988.

16. D. L. Pogoda. *Multiple Model Adaptive Controller For The STOL F-15 With Sensor/Actuator Failures.* Master's thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1988.

17. M.S. Sobota. *LQG/LTR Digital Control Law Design of a Robust Lateral Directional CGT/PI/KF Flight Controller for a STOL F-15 in a Landing Configuration.* Master's thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1986.

18. Warren Weinstein, et.al. Control Reconfigurable Combat Aircraft Development Phase 1 - R and D Design Evaluation. May 1987. AFWAL-TR-87-3011, Wright Patterson AFB, OH.

Vita

Captain Steven S. Payson attended the United States Air Force Academy from June 1980 to May 1984. He received a Bachelor of Science in Engineering. Captain Payson was then assigned to the Ballistic Missile Office at Norton AFB, where he was a Guidance and Control Project Officer working on the Small ICBM program. His present assignment is at the Air Force Institute of Technology in a master's program. Following graduation, Captain Payson will be reassigned to the Foreign Technology Division.

[redacted]

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188	
1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE			
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GE/ENG/89M-6		5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION School of Engineering	6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION	
6c. ADDRESS (City, State, and ZIP Code) Air Force Institue of Technology Wright-Patterson AFB Ohio 45433		7b. ADDRESS (City, State, and ZIP Code)	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AFWAL Flight Dynamics Lab	8b. OFFICE SYMBOL (If applicable) AFWAL/FDCLA	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS (City, State, and ZIP Code) Wright Patterson AFB Ohio 45433		10. SOURCE OF FUNDING NUMBERS	

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO	WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Flight Control System for the CRCA Using A Command Generator Tracker and Kalman Filter Volume II

12. PERSONAL AUTHOR(S)
Steven S. Payson, Captain, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Thesis	FROM ______ TO ______		

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Command Generator Tracker, Kalman Filter, Computer Aided Design, Linear Quadratic Gaussian, Flight Control Systems
01	04		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Thesis Advisor: Dr Peter S. Maybeck

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT		21. ABSTRACT SECURITY CLASSIFICATION	
[✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS		UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Peter S. Maybeck, Professor		22b TELEPHONE (Include Area Code) (513) 255-2057	22c OFFICE SYMBOL AFIT/ENG

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE

This research develops an integrated software design package useful in the synthesis of CGT/PI/KF control systems, and uses this software package to design and evaluate a longitudinal flight control system for the Control Reconfigurable Combat Aircraft (CRCA). The software package, called CGTPIKF and built with MATRIXX commands, allows for the synthesis and evaluation of a Command Generator Tracker (CGT) which provides inputs to the system and acts as a precompensator, and a regulator with proportional plus integral (PI) feedback which forces the system outputs to mimic the model output. The software also allows the incorporation of a Kalman filter for estimation of the system states. Certainty equivalence can be invoked by adopting the LQG assumptions, thereby allowing the Kalman filter to be designed independently of the CGT/PI controller. The total CGT/PI/KF controller can then be evaluated and the design refined. CGTPIKF is an interactive, menu driven CAD package which can be used in the development of any CGT/PI/KF control system, regardless of application.

A flight control system was designed for the CRCA air combat mode (ACM) entry using CGTPIKF. This control system was designed to force the aircraft to emulate a first order response in pitch rate. The command model of the command generator tracker represented a first order pitch rate response with a rise time of .6 sec. Various weighting matrices were evaluated and refined in the development of the PI controller; the different controller designs were tested against the simulation containing various modelling errors, particularly failure conditions. The Kalman filter was later added, and the controller was again tested against the failure conditions. Loop Transmission Recovery (LTR) was successfully implemented to enhance robustness. The results confirm that a robust control system can be designed using the software package developed in this research.